НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Система безпеки будинку з фото- та відеофіксацією»

Виконав:
студент IV курсу, групи ІА-61
Ткаченко Максим Сергійович _____

Керівник:
к.т.н., доцент
Креденцар Світлана Максимівна _____

Рецензент:
к.т.н., доцент
Лобанчикова Надія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Ткаченку Максиму Сергійовичу

1. Тема проєкту «Система безпеки будинку з фото- та відеофіксацією», керівник проєкту Креденцар Світлана Максимівна, к.т.н., доцент, затверджені наказом по університету від «7» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 9 червня 2020 р.

3. Вихідні дані до проєкту операційна система Android, мікрокомп'ютер Rasberry Pi, інфрачервоний датчик руху, камера відеоспостереження, хмарний сервіс для збереження та синхронізації даних

4. Зміст пояснювальної записки вступ, опис предметної області, структура та опис компонентів системи, розробка програмного забезпечення, особливості функціонування створеної системи, висновки, список використаних джерел

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) схема електрична функціональна, структурна схема системи, діаграма потоків даних, діаграма класів мобільного застосунку

6. Дата видачі завдання 6 березня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз існуючих рішень	09.03.20 - 17.03.20	
2	Розробка структурної схеми системи	18.03.20 - 22.03.20	
3	Розробка електричної функціональної	23.03.20 - 02.04.20	
	схеми		
4	Вибір та налаштування компонентів	03.04.20 - 14.04.20	
5	Розробка програми мікрокомп'ютера	15.04.20 - 30.04.20	
6	Реалізація функціоналу	01.05.20 - 14.05.20	
	мобільного застосунку користувача		
7	Налаштування сервісу синхронізації даних	15.05.20 - 22.05.20	
8	Розробка діаграми потоків даних	23.05.20 - 25.05.20	
9	Оформлення текстової документації	26.05.20 - 07.06.20	

Студент _____ Максим ТКАЧЕНКО

Керівник _____ Світлана КРЕДЕНЦАР

АНОТАЦІЯ

Ткаченко М.С. Система безпеки будинку з фото- та відеофіксацією. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 61 с. тексту, 23 рисунки, 1 таблицю, посилання на 28 літературних джерел та 4 додатки.

Ключові слова: система безпеки, відеоспостереження, розумний будинок, мобільний застосунок, віддалене керування, Android, Rasberry Pi, апаратна платформа.

Об'єктом розробки є система безпеки будинку з фото- та відеофіксацією.

Мета розробки — створення системи безпеки житлового будинку, яка здатна в режимі реального часу інформувати користувачів про потенційно небезпечні ситуації.

У дипломному проєкті розроблено систему безпеки житлового будинку, до складу якої входять наступні компоненти: мікрокомп'ютер, камера відеоспостереження, інфрачервоний датчик руху, мобільний застосунок користувача та хмарний сервіс синхронізації даних. Система може керуватися дистанційно через мобільний телефон з операційною системою Android. Обмін даними між апаратними платформами здійснюється завдяки підключенню до мережі Інтернет.

Розроблений прототип може бути використаний в якості приватної охоронної системи житлового будинку.

SUMMARY

Tkachenko M.S. Home security system with photo and video recording. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2020.

The project contains 61 p. of text, 23 images, 1 tables, 28 references and 4 additions.

Keywords: security system, video monitoring, smart home, mobile application, remote control, Android, Rasberry Pi, hardware platform.

The object of development is a home security system with photo and video recording.

The purpose of the development is creation of a security system for a residential building, which is able to inform users about potentially dangerous situations in real time.

The graduation project developed a security system for a residential building, which includes following components: microcomputer, a video camera, an infrared motion sensor, mobile user application and a cloud data synchronization service. The system can be controlled remotely via a mobile phone with the Android operating system. Data exchange between hardware platforms is carried out by connection to the Internet.

The developed prototype can be used as a private security system of a residential building.

Номер рядка	Формат	Позначення	Найменування	К-ть листів	Номер екземп.	Приміт-ки
1			Документація загальна			
2						
3			Знову розроблена			
4	А4	ІА61.250БАК.005 ПЗ	Пояснювальна записка	71		
5	А3	ІА61.250БАК.005 Д1	Система безпеки будинку з	1		
6			фото- та відеофіксацією.			
7			Схема підключення			
8			пристроїв мікрокомп’ютера			
9	А3	ІА61.250БАК.005 Д2	Система безпеки будинку з	1		
10			фото- та відеофіксацією.			
11			Структурна схема системи			
12	А3	ІА61.250БАК.005 Д3	Система безпеки будинку з	1		
13			фото- та відеофіксацією.			
14			Діаграма потоків даних			
15	А3	ІА61.250БАК.005 Д4	Система безпеки будинку з	1		
16			фото- та відеофіксацією.			
17			Діаграма класів мобільного			
18			застосунку			
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Лист	№ докум.	Підпис	Дата	ІА61.250БАК.005 ТП			
Розробив		Ткаченко М.С.			Система безпеки будинку з фото- та відеофіксацією Відомість проєкту	Літ.	Лист	Листів
Перевірив		Креденцар С.М.				Т		1
Н. контр.						КПІ ім. Ігоря Сікорського ФІОТ, Група ІА-61		
Затв.								

Пояснювальна записка
до дипломного проєкту
на тему: «Система безпеки будинку з фото- та відеофіксацією»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АП — апаратна платформа

БД — база даних

ОЗУ — оперативний запам'ятовуючий пристрій

ОС — операційна система

СУБД — система управління базами даних

УАПП — універсальний асинхронний приймач/передавач

CM — Cloud Messaging (технологія відправки повідомлень)

GPIO — General-purpose input/output (виходи загального призначення на платі мікрокомп'ютера для підключення пристроїв)

I2C — Inter-Integrated Circuit (інтерфейс послідовної асиметричної шини для зв'язку між інтегральними схемами)

IDE — Integrated Development Environment (інтегроване середовище розробки)

SDK — Software Development Kit (набір інструментів розробки програмного забезпечення)

SPI — Serial Peripheral Interface (послідовний периферійний інтерфейс передачі даних)

ВСТУП

В сучасному житті досить популярними стали різні автономні системи будинків. «Розумне світло», мікроклімат, голосове управління, економія електричної енергії — все це приклади таких систем. Вони мають досить широкий діапазон застосування: від систем для забезпечення комфорту до нових технологій, головним призначенням яких є, наприклад, заощадження коштів на комунальні витрати. Завдяки автоматизації, електроенергія, вода і тепло споживаються максимально економно, у деяких випадках збереження витрат може досягати 40-50%.

Розвиток сучасної мікропроцесорної техніки та безпровідних мереж, зменшення цін на електроенергію, поява нових інтелектуальних побутових приладів та мультимедійних технологій керування ними — саме ці фактори надають можливість створювати або легко розширювати такі системи. Це здійснюється шляхом поєднання абсолютно різних пристроїв в будинку під управлінням виділених центрів керування і синхронізації.

Об'єкт дослідження — це приватні системи безпеки, які створені для охорони житлових будинків, квартир, промислових або комерційних приміщень. Існуючі рішення таких систем легко інтегруються з мобільними пристроями, які дозволяють керувати налаштуваннями та отримувати миттєві сповіщення у випадку тривоги.

Предметом дослідження є можливість реалізації власної системи безпеки на основі сучасних апаратних платформ та телекомунікаційних можливостей.

Мета дипломного проєкту — створення охоронної системи для житлового будинку, яка здатна реагувати на можливі події тривоги в приміщенні (рух, відкриття дверей, відключення зв'язку) та інформувати користувачів в режимі реального часу про такі події.

Для створення системи використано наступні засоби програмування та платформи: мікрокомп'ютер Rasberry Pi, камера відеоспоспостереження,

інфрачервоний датчик руху, АП з операційними системами Android та Raspbian, сервіс Firebase в якості реалізації виділеного серверу в архітектурі системи. Весь обмін інформацією та авторизований доступ здійснюється через підключення до мережі Інтернет. Завдяки цьому, можливе дистанційне керування системою через мобільний телефон з ОС Android.

Результатом написання дипломного проєкту є прототип системи безпеки будинку, до складу якої входить налаштований мікрокомп'ютер з його периферійними пристроями та програмним кодом, мобільний застосунок користувача для керування і отримання сповіщень, а також сервіс збереження файлів та синхронізації даних між використаними платформами.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Наукові поняття «домашня автоматизація» або «розумний будинок» все більше стають реальністю, аніж абстрактними визначеннями. В таких будинках майже все контролюється автономно, від споживання електричної енергії до підтримки заданої температури приміщень, встановлення режиму охорони тощо. Приклади реалізації таких проєктів демонструють, що ідея автоматизації домашніх процесів у будинку надає широкі можливості для забезпечення комфорту та безпеки його мешканців.

Безліч пристроїв і систем у «розумних будинках» керуються дистанційно саме через мобільний телефон або планшет, на якому встановлено необхідне програмне забезпечення від виробника. Такий підхід є зручним і максимально простим для мешканців, адже з використанням лише одного пристрою можна контролювати різні сфери свого життя. Сучасна концепція розумного будинку зображена на рисунку 1.1 [1]:

Рисунок 1.1 — Сучасна концепція «розумного будинку», яка включає в себе велику кількість взаємопов'язаних систем

Доступ до функцій управління «розумним будинком» в програмному забезпеченні обов'язково потребує авторизації. Саме тому, підключення до таких автономних систем будинку здійснюється за допомогою безпечних протоколів в мережі Інтернет або безпосередньо підключенням до керуючого

комп'ютера з використанням локальної, захищеної мережі Wi-Fi. В свою чергу, виконавчі пристрої, центри керування та сенсори моніторингу параметрів утворюють також окрему, досить складну комп'ютерну мережу для синхронізації даних [2-3].

Схожі системи все більше і більше проникають в життя сучасних будинків, адже їх встановлення займає лише кілька годин, вони є простими і зручними у користуванні, а деякі з них навіть заощаджують витрати на комунальні платежі.

Проєктуванням і будівництвом «розумних будинків», а також створенням відповідного програмного забезпечення для них займаються спеціалізовані компанії.

1.1 Призначення та область застосування

Сучасні приватні системи безпеки, які є об'єктом дослідження дипломного проєкту, можуть бути використані для охорони житлових будинків, квартир, промислових приміщень тощо. Найголовніша функція таких систем — вчасне сповіщення власників будинку або мешканців у випадку потенційно небезпечної ситуації. Відповідно, це дає можливість оцінити ситуацію, вчасно зреагувати і як результат — уникнути небажаних наслідків.

1.2 Аналіз існуючих рішень

Варто сказати, що схожі системи, насправді, існують і входять до складу більш складних охоронних систем сучасних будинків, комерційних приміщень або підприємств. Деякі з них навіть встановлюються у великих житлових комплексах.

Популярним виробником систем для «розумного будинку» є вітчизняна компанія Ajax Systems, яка розробляє дизайн, апаратне і програмне

забезпечення для власних пристроїв. Вони створюють системи з акцентом на безпеку приміщень і вчасне інформування у випадку тривожних ситуацій. Як готовий продукт для порівняння, можна взяти комплект охоронної системи дистанційного керування «Ajax Starter Kit Cam». Зовнішній вигляд пристроїв зображено на рисунку 1.2 [4]:

Рисунок 1.2 — Охоронна система «Ajax Starter Kit Cam»

Головним в даній системі є керуючий пристрій (Hub), який під'єднаний до мережі через Ethernet підключення. Великою перевагою можна назвати наявність резервного, незалежного каналу зв'язку GSM (звичайна мобільна телефонія). Керуючий пристрій підтримує зв'язок із сервісами компанії, завдяки чому системою можна керувати дистанційно через програмне забезпечення, встановлене на мобільному телефоні. Система також може підняти тривогу у випадку втрати зв'язку із хмарним сервісом. Варто виділити ще одну перевагу — наявність окремої батареї живлення, яка може забезпечити автономність роботи пристроїв до 16 годин при відключенні від звичайної електричної мережі.

За наявності руху в приміщенні, система відразу відправляє на телефон push-повідомлення або SMS, також в залежності від налаштувань може

здійснити голосовий дзвінок на телефон користувача. Підключення мобільного застосунку здійснюється через сканування QR-коду, а сам застосунок дозволяє створювати групи охорони, додавати користувачів і змінювати права доступу, переглядати історію подій, керувати режимами охорони. Для оцінки ситуації, система надсилає серію фотографій менше ніж за 9 секунд.

Даний продукт є досить хорошим рішенням питання приватної охоронної системи, оскільки є простим у встановленні та подальшому користуванні, працює по зашифрованим протоколам передачі даних, містить резервні канали зв'язку для підвищення надійності і може бути легко розширена іншими пристроями від виробника Ajax Systems.

Існують також рішення для встановлення поза межами приміщень, на відкритому просторі. Наприклад, система GSM-сигналізації з назвою «GSM 10GA», яка містить в собі зовнішні інфрачервоні бар'єри для контролю окремих зон. Комплект охоронної системи «GSM 10GA» зображено на рисунку 1.3 [5]:

Рисунок 1.3 — Охоронна система «GSM 10GA» з інфрачервоними бар'єрами для контролю зон

При сигналі тривоги, система може здійснити голосовий дзвінок на різні мобільні номери (6 — максимальна кількість) або відправити до 3-х SMS-повідомлень. Містить вбудований акумулятор, з автономною роботою до 5 годин після відключення живлення електромережі.

Керування системою також можливе через мобільний застосунок на телефоні, який працює під операційними системами Android або iOS, а ще через спеціальні пульти управління, що постачаються в комплекті.

Можна сказати, що для охорони промислових приміщень це хороше рішення, але єдиний GSM-канал зв'язку, відсутність відеокамери спостереження та можливостей для розширення системи роблять її не самим надійним і привабливим продуктом серед конкурентів на ринку сучасних охоронних систем. Слід зауважити, що вартість цього комплекту орієнтовно така сама, як і попереднього від компанії Ajax Systems.

Варто розглянути ще один продукт від українського виробника Clever Apartment (скорочена назва — «CLAP»). Це досить молода компанія, яка займається розробкою системи «розумний будинок» для сучасних житлових комплексів.

На даний момент, введено в експлуатацію багато житлових комплексів, в яких працює система CLAP. В квартирах встановлюються датчики відкриття дверей та вікон, протипожежна сигналізація, датчики переповнення води, IP-камери відеоспостереження та інші пристрої для комфорту та економії електричної енергії. Система в режимі реального часу інформує користувачів про зміни в квартирі, або, наприклад, робить переадресацію з домофону на мобільний застосунок, що є досить зручним у ситуаціях, коли мешканців в квартирі немає. У випадку тривоги мешканці будуть інформовані повідомленнями і можуть здійснювати моніторинг стану квартири через телефон з використанням камер відеоспостереження. При відповідних налаштуваннях, система може сама викликати службу охорони.

На основі всіх описаних варіантів уже існуючих систем можна провести їх аналіз та здійснити порівняння функціоналу. Окрім параметрів надійності

та функцій забезпечення охорони, є доцільним вказати ціни існуючих на ринку систем, а також важливим параметром можна назвати здатність системи до розширення, оскільки сучасні «розумні будинки» можуть включати в себе велику кількість приміщень і, відповідно, потребують великої кількості датчиків та виконавчих пристроїв, які мають легко інтегруватися в систему під управлінням головного пристрою синхронізації та керування. Порівняння характеристик наведено в таблиці 1.1:

Таблиця 1.1 — Порівняння характеристик існуючих рішень охоронних систем

Існуючі системи / Критерії порівняння	Ajax Starter Kit Cam	GSM-сигналізація «GSM 10GA»	Система управління CLAP
Наявність власного живлення для автономної роботи	+	+	+
Резервні канали зв'язку	+	-	+
Камера відеоспостереження	+	-	+
Контроль великих зон поза приміщенням	-	+	-
Можливість розширення системи	+	-	+
Орієнтовна вартість на ринку	~ 280$	~ 220$	Залежить від оснащення будинку

За результатами аналізу існуючих рішень можна зробити певні висновки відносно їх функціоналу та технічних можливостей. Далеко не останнім фактором порівняння є ціна, і тут не завжди можливості системи залежать від вартості. Наприклад, деякі системи не можуть бути розширені,

але є більш надійними з точки зору безпеки. Інші — навпаки, легко інтегруються з додатковими пристроями, але поступаються в плані захищеності даних, що передаються. Взагалі, здатність будь-якої системи розширюватися і, відповідно, збільшувати можливості її використання — це один із найважливіших параметрів при проєктуванні, адже така необхідність допускається у майбутньому і має бути закладена відповідна технологічна основа.

Також варто сказати, що не всі із перевищених систем є сертифікованими, а тому питання виклику охоронної служби швидкого реагування в таких системах покладається лише на користувача. Системи від виробника Ajax мають постійну підтримку (періодичне збереження даних на стороні сервера, оновлення програмного забезпечення тощо), такий підхід, звісно, є більш технологічним серед представлених конкурентів. Хоча все це залежить від умов експлуатації та потреб користувача: інколи краще обрати систему більш надійну, без зайвих програмних або апаратних модулів.

В дипломному проєкті розробляється система безпеки з фото- та відеофіксацією, оскільки наявність камери відеоспостереження в наші дні можна назвати більше необхідністю, аніж додатковим функціоналом. Отриманий від системи медіафайл (фото чи відео) дає змогу оцінити відповідну ситуацію і вчасно прийняти рішення. Використання АП Rasberry Pi та периферійних пристроїв від цього виробника є значно дешевшим для створення власної системи безпеки, якщо порівнювати з існуючими продуктами.

Великою перевагою для систем безпеки є резервні канали зв'язку, адже, наприклад, доступ до мережі Інтернет може бути відключений і тоді користувач не отримає сповіщення. Для уникнення таких ситуацій деякі виробники додатково використовують мобільну телефонію (дзвінки та повідомлення) і тим самим підвищують надійність своїх систем. АП Rasberry Pi не дає можливості використати телефонію, тому єдиним каналом зв'язку у системі, що проєктується, виступає мережа Інтернет.

1.3 Постановка задачі дипломного проєктування

На основі аналізу предметної області та характеристик існуючих рішень, виявлення їх недоліків і переваг, порівняння вартості систем можна поставити наступну задачу дипломного проєктування. Необхідно реалізувати власну систему безпеки з фото- та відеофіксацією, яка буде керуватися дистанційно через мобільний застосунок на телефоні та відправляти користувачам сповіщення у випадку тривожних ситуацій.

Окремо можна виділити такі завдання дипломного проєктування:

— реалізувати методи комунікації між різними платформами (синхронізація налаштувань, обмін даними, завантаження файлів);

— написати мобільний застосунок користувача, програму мікрокомп'ютера та функції для обробки даних на сервері;

— мінімізувати час затримки на завантаження файлів і синхронізацію даних з метою вчасного інформування користувачів;

— зробити систему простою для встановлення у будь-якому приміщенні та легкою для подальшого використання. Врахувати вартість обраних пристроїв та можливість розширення системи на основі існуючого функціоналу.

2 СТРУКТУРА ТА ОПИС КОМПОНЕНТІВ СИСТЕМИ

Створена в дипломному проєкті система є інтеграцію різних платформ, програмного забезпечення та фізичних пристроїв, які можуть здійснювати комунікацію між собою через мережу Інтернет.

2.1 Головні компоненти архітектури

Серед архітектурних компонентів системи варто виділити головні: мобільний застосунок користувача, мікрокомп'ютер та хмарний сервіс для комунікації і збереження даних. База даних і сховище файлів розділені між собою, між ними лише асоціативний зв'язок, оскільки записи в БД містять посилання на файли, збережені в хмарному сховищі. Спрощену структурну схему системи зображено на рисунку 2.1:

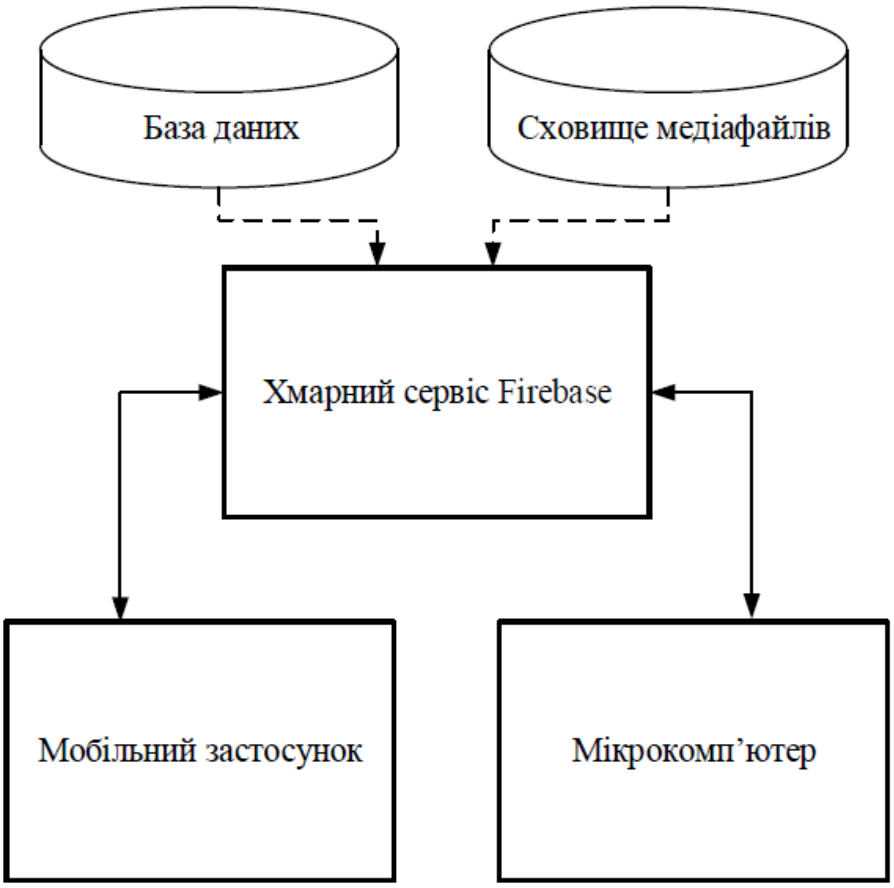


Рисунок 2.1 — Спрощена структурна схема системи

На кресленику ІА61.250БАК.005 Д2, який відображає розгорнуту структурну схему системи, кожен компонент архітектури містить в собі кілька інших складових. Наприклад, мобільний застосунок складається з екранів графічного інтерфейсу, локальної бази даних, служби синхронізації повідомлень тощо. Сервіс збереження даних містить в собі окрему БД та сховище медіафайлів, які не пов'язані між собою і є окремими посиланнями для запитів зі сторони клієнта.

Мобільний застосунок можна встановити на телефон з ОС Android, він є головним засобом керування системою для користувача. Складається зі служб комунікації та обміну даними, локальної БД та екранів графічного інтерфейсу. На іншій стороні системи (наприклад, в приміщенні будинку) встановлений мікрокомп'ютер, який є незалежним і автономним, має власне живлення і може керуватися дистанційно. До нього підключені периферійні пристрої: датчик руху та камера відеоспостереження. На мікрокомп'ютері в циклічному режимі виконується програма, яка зчитує покази від датчика, керує камерою, може вносити зміни до бази даних на хмарному сервісі або завантажувати файли до сховища. Якщо датчик реагує на рух в приміщенні, програма миттєво вмикає камеру відеоспостереження, робить фото або відео і відразу відправляє сповіщення власнику на мобільний телефон з отриманим файлом. Відбувається це з мінімальною затримкою в часі.

Мікрокомп'ютер не може взаємодіями безпосередньо з мобільним застосунком. З метою налаштування комунікації між платформами використано хмарний сервіс. Функції синхронізації виконують роль виділеного сервера в архітектурі системи і відповідають за відправку повідомлень на мобільний телефон, передачу даних від клієнта до програми мікрокомп'ютера тощо. Це певний перелік налаштувань і правил всередині самої бази даних. Якщо програма мікрокомп'ютера робить новий запис (фіксує подію в приміщенні), одна із функцій синхронізації бере дані про цю подію з відповідної таблиці, шукає посилання на медіафайл і потім відправляє push-повідомлення на сторону клієнта.

2.2 Мобільний застосунок користувача

Застосунок встановлюється на мобільний телефон і надає користувачу можливість керувати системою та отримувати сповіщення. Обов'язковою умовою використання є авторизація користувача в системі. Після входу або створення нового профілю, доступний весь функціонал застосунку. Серед основних функцій можна назвати такі:

— відключення режиму охорони або його активація;

— отримання сповіщень у випадку потенційно небезпечних ситуацій, які зафіксував мікрокомп'ютер у приміщенні;

— запит на отримання фото чи відео в режимі реального часу для оцінки ситуації;

— перегляд медіафайлів та можливість їх видалення із хмарного сховища;

— зміна налаштувань системи (тривалість запису відео або тип медіафайлу, який відправляє система).

Написаний мобільний застосунок містить в собі сервіс, який працює в окремому потоці, не залежить від стану програми і відповідає за інформування у випадку тривоги. Це означає, що користувач отримає повідомлення навіть у той момент, коли не взаємодіє безпосередньо з програмою.

2.3 Сервіс збереження та синхронізації даних

Оскільки в дипломному проєкті використано кілька апаратних платформ, на яких встановлені різні операційні системи, необхідним є створення засобу комунікації та синхронізації даних між цими платформами. Проблема вибору готового або створення власного сервісу виникла на самому початковому етапі проєктування. В архітектурі системи немає виділеного серверу як фізичного пристрою, тому ідеальним рішенням для забезпечення

потреб системи є використання хмарної технології. Такий підхід надає можливість легко реалізувати обмін повідомленнями між клієнтами в режимі реального часу, що є необхідністю в системах охорони і безпеки.

Для відправки повідомлень, збереження файлів і записів в базі даних обрано досить розвинену хмарну технологію — сервіс Firebase. Даний сервіс дозволяє створювати якісні програми, збільшувати аудиторію користувачів через підключення аналітики, створювати серверну частину для мобільних застосунків. Інакше кажучи, це інтуїтивно зрозумілий програмний засіб, який бере на себе роль керування окремими пакетами мережевого трафіку. Завдяки готовій інфраструктурі не потрібно працювати з великою кількістю налаштувань, можна сконцентруватися на потребах користувачів [6-7]. Сервіс має безкоштовну підтримку і є крос-платформним, може легко масштабуватися у випадку, коли створений проект стане популярним і навантаження на нього буде зростати. При такому сценарії, не потрібно змінювати налаштування серверу, в залежності від необхідних ресурсів для роботи проекту буде змінено тарифний план розробника. Також великою перевагою використання сервісу є можливість підключення безпечної авторизації через відкритий протокол OAuth 2.0, завдяки якому надається обмежений доступ до ресурсів без необхідності передавати стороннім службам логін або пароль користувача [8].

Серед всіх функцій, які надає Firebase розробникам для швидкого розгортання власних системи, варто звернути уваги на такі:

— власна СУБД типу NoSQL, яка дозволяє зберігати і синхронізувати дані між клієнтами;

— підтримка інтеграції з мобільними застосунками під операційні системи iOS/Android;

— збереження файлів і забезпечення авторизованого доступу;

— наявність сервісу для відправки миттєвих повідомлень на мобільні пристрої;

— можливість інтеграції зі службами аналітики та збору даних;

— можливість створення власних функцій управління даними та комунікації з клієнтами без використання окремих виділених серверів.

Створена в дипломному проєкті система використовує більшість із цих функцій для забезпечення своєї роботи. База даних використовується для збереження історії подій, які фіксує системи, налаштувань користувача, даних синхронізації і управління мікрокомп'ютером, зберігає користувачів і їх унікальні ідентифікатори для авторизації і доступу до файлів.

Також можна виділити ще один фактор, який є досить вагомим при виборі саме сервісу Firebase, а не іншого серед існуючих на даний момент. Інтеграція функцій Firebase є досить зручною безпосередньо з середовища Android Studio IDE. Ця легкість використання для старту розробки пояснюється тим, що і ОС Android, і середовище розробки мобільних застосунків для неї, і сам сервіс Firebase підтримує одна компанія — Google, яка втілює у своїх продуктах і сервісах зручність, доступність і простоту використання не лише користувачами, а й розробниками програмного забезпечення.

2.4 Мікрокомп'ютер

В дипломному проєкті використано АП Rasberry Pi. Це сімейство компактних одноплатних мікрокомп'ютерів, головним покликанням яких є орієнтація на навчання та експерименти. Варто сказати, що завдяки широкому діапазону можливостей і низькій вартості популярність платформи перевищила очікування виробників з моменту її появи на світовому ринку і зацікавила велику аудиторію ентузіастів в сфері створення систем автоматизації. АП легко інтегрується з більшістю виконавчих пристроїв, сенсорів, а деякі модулі (наприклад, камера відеоспостереження) розробляються самим виробником Універсальність та відкритість проекту призвела до того, що ряд комерційних продуктів було створено і випущено на ринок саме на платформі Rasberry Pi.

2.4.1 Загальні відомості

На даний момент, існує 4 покоління мікрокомп'ютерів Rasberry Pi, кожне з яких має власні модифікації. Платформи різних поколінь відрізняються наявністю тих чи інших портів або пристроїв на платі, об'ємом ОЗУ тощо. При написанні дипломного проєкту використаний мікрокомп'ютер 3 покоління Rasberry Pi Model 3B+ , його зовнішній вигляд зображено на рисунку 2.2 [9]:

Рисунок 2.2 — Мікрокомп'ютер Rasberry Pi Model 3B+

Великою перевагою платформи є наявність низькорівневих інтерфейсів, які дозволяють підключати до плати розширення, зовнішні контролери, виконавчі пристрої, датчики тощо. Серед недоліків можна назвати неповний доступ до цифрового сигнального процесору (DSP), відсутність власного годинника реального часу (єдиний спосіб отримати актуальний час — синхронізація із зовнішніми серверами), а також можливість легко пошкодити апаратні модулі, наприклад через незахищеність портів виводу від короткого замикання (при неправильному підключення платформа може сама собі заподіяти шкоду). Ще один недолік — робота мікрокомп'ютера лише з цифровими сигналами. Відповідно до цього, будь-який аналоговий сигнал на

вході або виході перетворюється в цифровий завдяки широтно-імпульсній модуляції [10].

2.4.2 Технічні характеристики

АП Rasberry Pi 3 покоління включає в себе 4-х ядерний процесор архітектури ARM із тактовою частотою 1.4 ГГц, графічний процесор, ОЗУ об'ємом 1 ГБ, безпровідні модулі Wi-Fi та Bluetooth. Серед портів підключення є аудіо- та відеоінтерфейс, окремий апаратний модуль для роботи з камерою відеоспостереження та виводом зображення на дисплей, 4 USB-порти, а також 3.5мм аудіоінтерфейс для підключення зовнішніх звукових пристроїв та вихід для кабелю Ethernet. Схема апаратних модулів мікрокомп'ютера Rasberry Pi Model 3B+ представлена на рисунку 2.3:

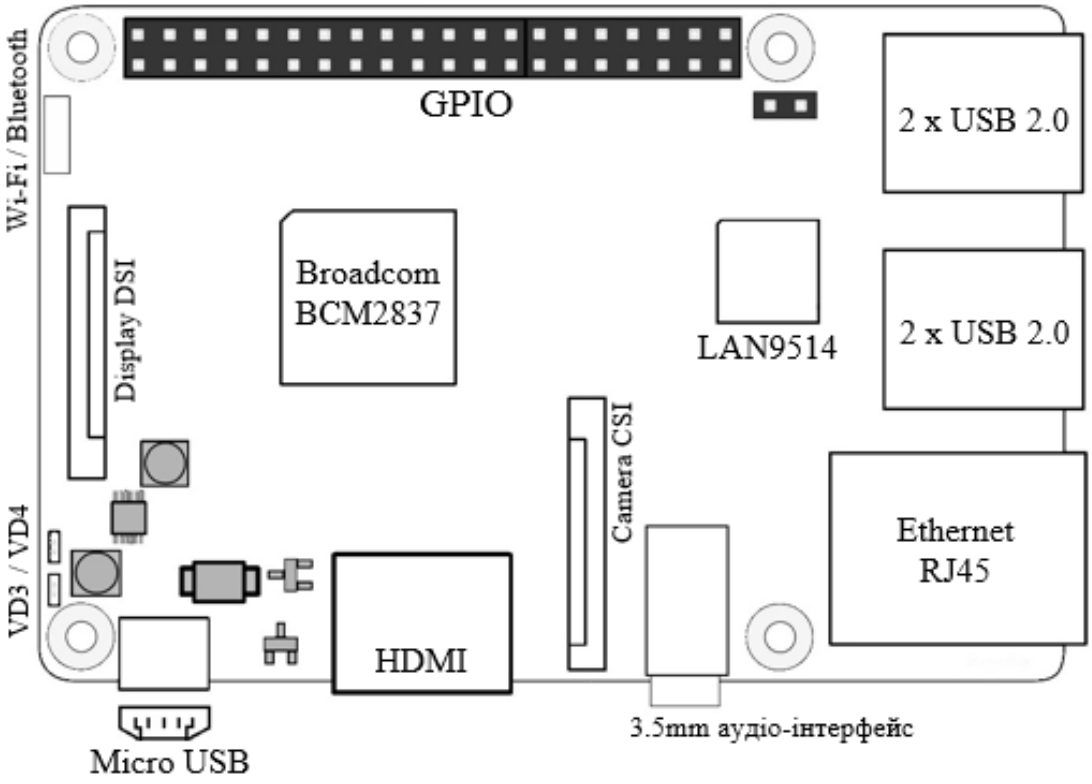


Рисунок 2.3 — Схема апаратних модулів мікрокомп'ютера

Власної пам'яті мікрокомп'ютер не має і використовує зовнішню карту пам'яті, на якій встановлений спеціалізований дистрибутив операційної

системи, створений на базі ядра Linux. Для роботи мікрокомп'ютер потребує певних периферійних пристроїв (блок живлення, кабель для підключення монітору тощо) [11].

Для живлення АП використовується звичайний вихід Micro USB. Модуль ОЗУ, графічний та основний процесори об'єднані в 1 елемент на платі, який позначений назвою Broadcom BCM2837. Управління периферійними пристроями через USB-порти та синхронізацію даних від Ethernet-підключення здійснює окремий мікроконтролер LAN9514. На платі є 2 світлові діоди, які слугують для інформування про наявність живлення та завантаження процесора.

Всі зовнішні датчики або виконавчі пристрої під'єднуються до виходів загального призначення GPIO на платі мікрокомп'ютера за допомогою звичайних проводів з'єднання. Розташування виходів GPIO зображено на рисунку 2.4 [12]:

Рисунок 2.4 — Виходи загального призначення GPIO на платі мікрокомп'ютера

Серед цих виходів є порти УАПП, інтерфейси I2C та SPI, виходи живлення 3.3В або 5В, а також порти загального призначення для програмування виконавчих пристроїв або зчитування інформації від сенсорів.

Нумерацію та розташування всіх портів мікрокомп'ютера зображено на рисунку 2.5 [13]:

Рисунок 2.5 — Нумерація та розташування портів мікрокомп'ютера

Порти загального призначення мають логічну та фізичну нумерацію. Для використання порту потрібно знати його призначення, логічний номер для ініціалізації в програмі та фізичний номер на платі для підключення виконавчих пристроїв або інформаційних каналів. Керувати портами можна через директорії і файли налаштувань, до яких можна звертатися для запису або зчитування. Це віртуальна файлова система, яку надає розробникам ядро Linux. Інший варіант доступу до портів введення/виведення — підключення системних бібліотек до програми.

Електронні комутатори та чіпи можуть бути пошкоджені через неправильно підключення, тому при використанні великої кількості портів обов'язково слід робити розв'язку через перетворювачі рівня напруги, електронні ключі тощо [14].

Також слід використовувати додаткові ключі на транзисторах або мікросхемах при підключенні периферійних пристроїв до портів живлення 3.3V або 5V. Якщо перевищити максимально можливе значення сили струму через підключення, буде перенавантаження для порту GPIO.

Сила струму, яку використовує вся платформа, залежить від навантаження на мікропроцесор та графічний процесор, кількості підключених виконавчих пристроїв через USB-порти та виходи загального призначення. Цей показник може змінюватися в діапазоні від 250мА до 1А.

Як і будь-яка чутлива електроніка, мікрокомп'ютер Rasberry Pi може бути пошкоджений статичним струмом, це також треба враховувати при експлуатації платформи.

На кресленику ІА61.250БАК.005 Д1 зображено периферійні пристрої мікрокомп'ютера, серед яких є перетворювач напруги. Він підключається до електромережі з напругою 220В та забезпечує живлення платформи через порт microUSB. За стандартом цього порту, сила струму становить 2.5А, напруга — 5В.

2.4.3 Операційна система

Варто зробити пояснення, що мікропомп'ютери сімейства Rasberry Pi працюють на базі власної операційної системи з назвою «Raspbian». Насправді, це не повністю унікальна розробка від виробника мікрокомп'ютерів, оскільки операційна система Raspbian є одним із багатьох дистрибутивів, створених на базі ядра операційної системи Linux. Від цього розробники отримують дуже багато переваг, адже, ще до початку написання власних програм, вже знайомі всі команди управління, інтерфейс користувача, файлова система.

При необхідності легко інтегруються пакети взаємодії з фізичними пристроями та модулями мікрокомп'ютера, сервіси та сторонні бібліотеки. ОС Raspbian дуже добре оптимізована під процесори ARM, які не відносяться до категорії самих потужних у своєму класі. ОС активно розвивається в наші дні завдяки співпраці між аудиторією користувачів та розробниками платформи.

2.5 Камера відеоспотереження

Для мікрокомп'ютерів сімейства Rasberry Pi існує 3 покоління камер відеоспостереження, які поставляються на ринок саме від офіційного виробника. В даному дипломному проєкті використана камера 2-го покоління

Rasberry Pi Camera Module V2, яка має досить високу роздільну здатність матриці (8 мегапікселів), може робити знімки з максимальним розміром 3280 x 2464 і здійснювати запис відео у стандартах якості 1080p30, 720p60 та 480p90 [15]. Зовнішній вигляд камери відеоспостереження та шлейфу для підключення зображено на рисунку 2.6 [16]:

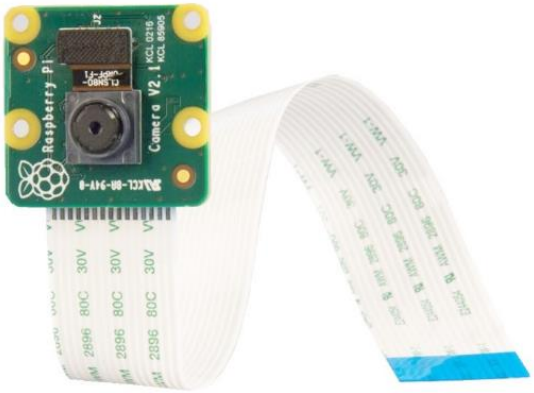


Рисунок 2.6 — Модуль камери Rasperry Pi 2-го покоління

2.5.1 Підключення та взаємодія з модулем

Варто зауважити, що дистрибутив ОС Raspbian вже містить в собі драйвери та всі необхідні методи для роботи з камерою в окремому пакеті системних бібліотек. В програмному коді можна динамічно змінювати налаштування камери, а всі медіафайли зберігаються у файловій системі. Активувати апаратний модуль для підключення камери відеоспостереження можна в налаштуваннях ОС (рисунок 2.7).

Рисунок 2.7 — Апаратний модуль для підключення камери відеоспостереження на платі мікрокомп'ютера

2.5.2 Формат стиснення відео

Відео записується у форматі стиснення «H.264» і для відтворення на більшості сучасних ОС через стандартні програми потребує додаткової конвертації в більш популярні, загальноприйняті формати, що створює певні незручності. Це викликано великим об'ємом відеоматеріалу в системах спостереження чи розпізнавання інформації, які найчастіше створюється з використанням даної платформи. Стандарт стиснення «H.264» забезпечує зменшення розміру вихідного файлу без втрат якості. Окрім цього, стиснення з використанням даного формату пояснюється слабким графічним процесором мікрокомп'ютера.

2.6 Датчик руху

Для виявлення руху в приміщенні використаний інфрачервоний датчик обходу перешкод. Чутливість сенсору в залежності від умов освітлення або відстані взаємодії з об'єктами руху може бути налаштована через відповідні потенціометри на платі. Максимальна відстань виявлення руху становить 2 метри. Датчик потребує живлення від 3 до 5В, має 1 інформаційний вихід для підключення до пристроїв управління та зчитування інформації. Зовнішній вигляд датчика зображено на рисунку 2.8 [17]:

Рисунок 2.8 — Інфрачервоний датчик руху

Основні елементи — це інфрачервоні світлові діоди. Один із них відправляє інформаційний сигнал, інший отримує відбитий від об'єктів інфрачервоний світловий промінь, саме так фіксується наявність предметів або перешкод в полі зору датчика. Підключення датчика руху до портів загального призначення GPIO зображено на кресленику «Система безпеки будинку з фото- та відеофіксацією. Схема підключення пристроїв мікрокомп'ютера».

На корпусі датчика присутні 2 потенціометри (B1, B2) для налаштувань чутливості світлових діодів, 2 світлові діоди-індикатори (живлення та інформаційний сигнал — VD1, VD2), а також логічний інтерфейс обробки сигналу. Для підключення до мікрокомп'ютера на платі позначено 3 виходи: VCC (живлення), GND (мінус), OUT (інформаційний сигнал). Схема апаратних модулів інфрачервоного датчика руху представлена на рисунку 2.9:

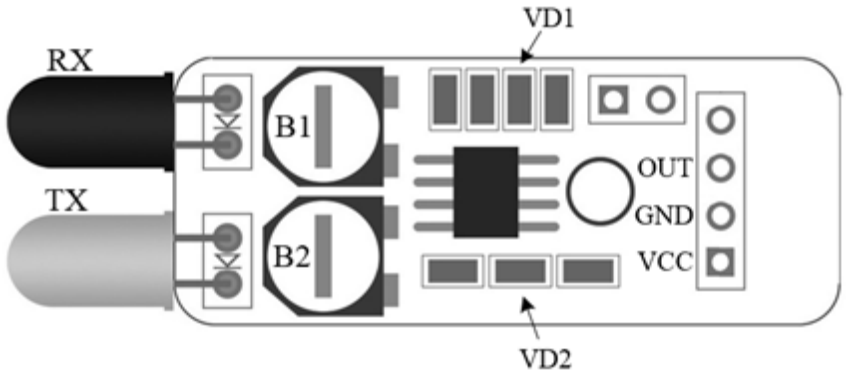


Рисунок 2.9 — Схема апаратних модулів інфрачервоного датчика руху

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення написано окремо для різних АП, а також для модулю системи, який відповідає за комунікацію між ними, відправку повідомлень та збереження файлів.

3.1 Розробка та опис мобільного застосунку

Створений в дипломному проєкті мобільний застосунок містить в собі значну кількість екранів графічного інтерфейсу і є основним засобом управління системою. Містить в собі програмний код реалізації фунціоналу системи, службу відправки повідомлень, налаштування компіляції під різні архітектури мобільних пристроїв, а також додаткові ресурси, серед яких окремо знаходяться зображення векторного формату, всі файли розмітки створеного графічного інтерфейсу, графи навігації між екранами тощо.

3.1.1 Середовище розробки

Мобільний застосунок користувача написано на базі операційної системи Android, з використанням спеціалізованого середовища Android Studio IDE. Це офіційний і повністю безкоштовний продукт, який активно розвивається в наші дні. Android Studio містить в собі безліч можливостей для написання програм не тільки під мобільні телефони, а навіть під сучасні «розумні годинники», або, наприклад, телевізори з підтримкою Android TV. Дозволяє легко підключати сторонні сервіси та бібліотеки до проєкту, а також надає можливості для створення графічного інтерфейсу, моніторингу стану пристрою під час виконання програми, пошуку помилок в програмному коді тощо.

Середовище Android Studio є досить гнучким, завдяки цьому процес створення програми адаптується під вимоги розробника. Можна спостерігати

за змінами в проекті, підключати системи контролю версій, працювати з відразу з багатьма репозиторіями.

В програму влаштований емулятор, який дозволяє перевірити коректність роботи мобільного застосунку на пристроях з різними екранами та апаратною частиною (архітектура процесора, об'єм ОЗУ). Це стало досить актуальним з появою великої кількості смартфонів та планшетів на ринку з ОС Android.

Також серед вбудованих в IDE інструментів є великий набір шаблонів для розробки програмного забезпечення, засоби тестування різних компонентів, утиліти для рефакторингу існуючого коду тощо.

3.1.2 Набір інструментів розробки Android SDK

Для написання мобільних застосунків під ОС Android існує універсальний набір інструментів Android SDK. Він дозволяє легко створювати і тестувати програми, а також оцінювати сумісність з різними версіями ОС. Більшість пакетів написані мовою програмування Java, серед яких можна виділити засоби розробки, набір бібліотек, телефонний емулятор, документацію та зразки використання тих чи інших програмних засобів. Android SDK може включати фрагменти застарілих версій Android, це необхідно у випадку підтримки раніше створених програм або коли розробники мають намір продовжити розвиток своїх застосунків для застарілих телефонів або планшетів.

Android SDK містить системні бібліотеки для роботи з різними апаратними пристроями сучасних мобільних девайсів (камера, акселерометр, компас, GPS, доступ до модулів Bluetooth та Wi-Fi тощо). Також є окремі пакети для роботи з браузером, телефонією, інтегрованими базами даних, мультимедійним контентом (фото та відео різних форматів).

Після компіляції у фінальному вигляді будь-який застосунок представляє собою набір пакетів («android package», файл типу «.apk»), який

може поширюватися та встановлюватися на пристрої з необхідною версією ОС. Всередині пакетів знаходяться файли для виконання на віртуальній машині, ресурси, файли базових налаштувань [18].

Оскільки ОС Android створена на базі ядра Linux, кожен застосунок в ній виконується у своєму процесі. Процес запускається, коли необхідно виконати певний компонент (фрагмент коду) мобільного застосунку, після цього ОС сама знищує процес. У кожного процесу є власна віртуальна машина. Саме завдяки такому підходу реалізовано «принцип надання мінімальних прав». Цей принцип означає, що кожен застосунок має доступ лише до необхідних для його роботи компонентів, а сама ОС Android гарантує цим безпечне середовище для користувача. Якщо необхідно отримати доступ до інших компонентів системи (наприклад, телефонія, камера, запис або читання файлів з пам'яті), користувач обов'язково повинен надати доступ мобільному застосунку при його встановленні [19].

3.1.3 Відомості про мову програмування Kotlin

Мобільний застосунок написано мовою програмування Kotlin. Варто зауважити, що для розробників є широкий вибір мов, які підтримує середовище розробки і які легко компілюються на мобільних телефонах. Це може бути, наприклад, Java або C++. В 2017 році компанія Google зробила оголошення відносно того, що Kotlin прийнято вважати офіційним інструментом розробки програм і мобільних застосунків на базі операційної системи Android. Це досить молода мова програмування, яка створена на базі віртуальної машини Java. Розробники мови Kotlin вирішили питання застарілості класичних високорівневих мов програмування, взявши від них найкращі підходи при проєктуванні і додали сучасні складні алгоритми, пов'язані, наприклад, з реактивним програмуванням, адаптивністю і компіляцією на різних архітектурах процесорів.

Для платформи Android мова Kotlin інтегрується через спеціальну

систему автоматичної компіляції під назвою Gradle. Через файли цієї системи можна підключати та видаляти бібліотеки, змінювати версії підтримки ОС, налаштовувати тестові варіанти компіляції тощо. Великою перевагою є те, що мову Kotlin можна підключати до уже існуючих проєктів і продовжувати розробку саме з її використанням, без повного переписування існуючого програмного коду.

Зараз представлено не так багато операційних системи, які є популярними на світовому ринку серед мобільних пристроїв. Якщо говорити про існування схожих підходів та засобів програмування, то варто згадати операційну систему iOS і таку саму сучасну мову програмування Swift, розробкою яких займається компанія Apple. Можна зробити висновок, що наявність власних інструментів розробки програмного забезпечення для компаній-гігантів є великою необхідністю, адже від цього залежить якість продукту, а значить і успіх його реалізації на світовому рівні.

3.1.4 Основні компоненти архітектури

Архітектура будь-якого мобільного застосунку включає в себе наявність екранів представлення (компонент системи Android з назвою «Activity») та нижчих по ієрархії фрагментів інтерфейсу (компонент системи Android з назвою «Fragment»), які відповідають за відображення інформації та взаємодію користувача з програмою. Якщо цю структура представити дещо простіше, то в сучасному застосунку будь-який інтерфейс — це фрагмент, а весь екран представлення містить в собі багато таких фрагментів та відповідає за навігацію між ними. При встановленні мобільного застосунку, кожна активність реєструється в ОС і може бути відкрита навіть через сторонні застосунки. Наприклад, це відбувається коли користувач хоче відправити файл або повідомлення, ОС Android в такому випадку пропонує вибір, через яку саме програму буде здійснено відправку повідомлення. На системному рівні всі мобільні застосунки мають ідентифікатор (назва пакету), і певні

активності налаштовані так, що вони доступні для відкриття ззовні. Попередньо треба вказати лише категорія та дію, яку виконує активність (наприклад, «action view», «action send», «action call» тощо).

Діаграма класів мобільного застосунку, зображена на кресленику ІА61.250БАК.005 Д4, ілюструє, що архітектура має велику кількість фрагментів, однак лише 2 активності:

— AuthActivity: відповідає за авторизацію користувача в системі, містить в собі екрани привітання, реєстрації, введення логіну з паролем.

— MainActivity: містить в собі головні екрани застосунку, такі як екран доступних функцій, архів медіафайлів, налаштування, перегляд фото чи відео, відображення повідомлень.

Це класичний підхід при проєктуванні, коли користувач може отримати доступ до головних функцій програми лише після авторизації. Інакше кажучи, всі екрани з основним функціоналом ніяк не можуть бути доступними для неавторизованого користувача. Після входу в систему на сервері створюється унікальний ідентифікатор для відправки повідомлень конкретному користувачу, цей ідентифікатор зберігається і в базі даних, і в локальному сховищі мобільного застосунку.

Також в архітектурі мобільного застосунку важливо виділити служби (компонент системи Android з назвою «Service»). Це додаткові служби, які можуть працювати в різних потоках і не залежать від того, чи відкритий зараз мобільний застосунок і чи використовує його власник пристрою. Такі служби працюють навіть після закриття мобільного застосунку і відповідають, наприклад, за оновлення геолокації, відображення щойно прийнятих повідомлень, планування задач на майбутнє тощо. Всі довготривалі операції, які виконуються у фоновому режимі і не містять інтерфейсу користувача слід виносити в окремі служби. Служби виконуються завдяки міжпроцесорній взаємодії і можуть викликатися різними компонентами ОС Android. Наприклад, служба може обробляти мережеві запити, відтворювати в

фоновому режимі медіафайл (музика або відео), виконувати завантаження файлів з мережі Інтернет тощо [20].

В мобільному застосунку існує 1 служба під назвою «CloudMessagingService» (лістинг коду — додаток Г), вона постійно тримає зв'язок із сервером і миттєво реагує на нові події, які зафіксувала система на стороні мікрокомп'ютера. Саме ця служба генерує push-повідомлення, яке надходить користувачу у випадку, коли в приміщенні зафіксовано рух. Служба зареєстрована в ОС, працює постійно і вимкнути її з точки зору безпеки неможливо.

Всі графічні ресурси в мобільному застосунку — це векторні зображення. Використання векторної графіки дає змогу зберігати високу якість інтерфейсу на мобільний пристроях з абсолютно різними екранами. Це стосується і файлів розмітки графічного інтерфейсу, вони описуються мовою «xml» і є незалежними від програмного коду. При необхідності створення графічного інтерфейсу, наприклад, під специфічні розміри екранів (наприклад, планшети), в проєкті можна створювати окремі директорії і вказувати, під який саме екран в них знаходяться файли розмітки графічного інтерфейсу. Звісно, це збільшує розмір мобільного застосунку, але робить його більш гнучким і оптимізованим під різні мобільні пристрої.

Сторонні пакети або бібліотеки підключаються через систему автоматизованого зібрання проекту Gradle. В цій системі можна налаштовувати функції тестування, вказувати налаштування компіляції тощо.

До створеного проєкту в середовищі Android Studio підключено багато сторонніх бібліотек, які також є компонентами архітектури застосунку. Серед них найголовніші:

— Android Core: бібліотека містить в собі всі системні класи для розробки мобільного застосунку, в тому числі служби, елементи графічного інтерфейсу, засоби для роботи з апаратними пристроями на мобільному телефоні, налаштування компіляції програмного коду, оптимізації під різні пристрої;

— Navigation: бібліотека для створення графів навігації між екранами інтерфейсу користувача. Дозволяє налаштовувати переходи, передавати певні аргументи або параметри між екранами, повертатися до раніше відкритих екранів тощо;

— Firebase messaging: бібліотека для інформування користувачів і відправки push-повідомлень. Відповідає також за розгортання служби комунікації на базі ОС та реалізацію обміну даними між клієнтом і сервером;

— Firebase database та Firebase storage: бібліотеки для роботи з базою даних (запис, зчитування, видалення даних з таблиць) та хмарним сервісом збереження файлів. Окрім функцій обміну та керування даними, ці бібліотеки забезпечують авторизований доступ до всіх записів на сервері;

— Gson: досить поширена бібліотека для різних ОС, відповідає за конвертацію даних, серіалізацію класів у проєкті, коректну роботу з моделями, що передаються через HTTP-протокол тощо;

— Glide: завантаження медіафайлів, їх відображення на мобільному пристрої;

— Hawk: одна із найрозвиненіших криптографічних бібліотек на мобільних телефонах. Дозволяє безпечно зберігати ідентифікатори доступу, паролі, дані користувача. Бібліотека використана в проєкті з метою збереження токенів авторизації та налаштувань системи.

Архітектурний шаблон мобільного застосунку — MVVM (Model-View-ViewModel). Даний шаблон орієнтований на сучасні платформи розробки. З самого початку шаблон був орієнтований на розробку програм під платформу Windows і лише в останні роки набув популярності серед мобільних платформ. Головним покликанням цього шаблону є розділення логіки та графічного інтерфейсу програм. Всі моделі і служби представлення даних в проєкті при використанні такого архітектурного шаблону є незалежними один від одного, що значно спрощує тестування програми в подальшому.

Патерн MVVM складається з 3-х компонентів: Model (представляє собою дані, які необхідно показати користувачу), View (класи, які

відповідають відображення даних та інтерактивість), ViewModel (компонент для забезпечення актуальності даних, які необхідні користувачу, а також для зміни стану програми в залежності від сценарію використання). Схему архітектурного шаблону MVVM зображено на рисунку 3.1:

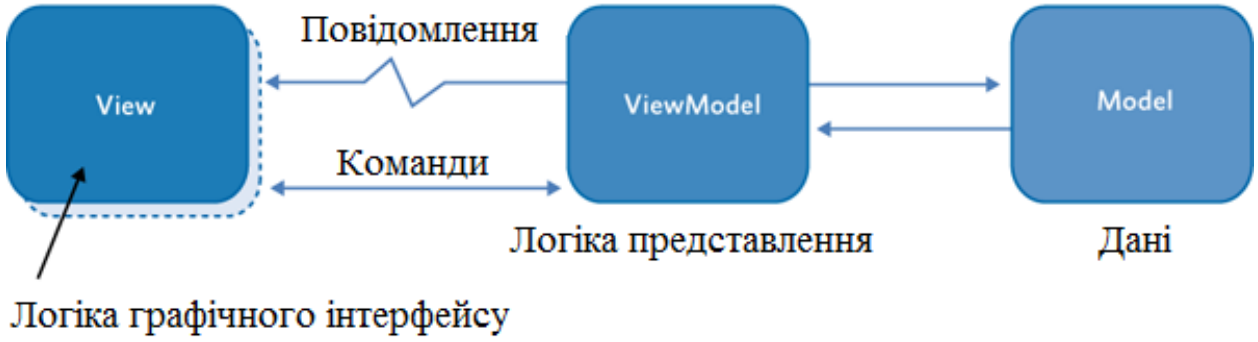


Рисунок 3.1 — Схема архітектурного шаблону MVVM

Результатом використання патерну MVVM є функціональне розділення системи на компоненти, які простіше розробляти, підтримувати, тестувати та модифікувати в майбутньому [21].

Для ідентифікації мобільного застосунку в операційній системі, його активностей, компонентів і служб є окремий файл в структурі проекту з назвою «AndroidManifest.xml». В ньому реєструється пакет застосунку (унікальний ідентифікатор), вказується орієнтація екранів та загальний стиль графічного інтерфейсу, реєструються всі активності, служби, а також постачальники контенту (компонент системи Android з назвою «ContentProvider», необхідний для забезпечення доступу до інформації іншим мобільним застосункам або службам). Якщо до мобільного застосунку підключаються сторонні сервіси (наприклад, сервіси відправки повідомлень, авторизації через соціальні мережі, робота з картами тощо), їх унікальні ідентифікатори і ключі доступу також потрібно обов'язково вказувати в даному файлі конфігурації. Кожен мобільний застосунок містить в собі такий перелік налаштувань і зареєстрованих компонентів в ОС Android. Ці базові налаштування мобільних застосунків зберігаються у відповідних системних

файлах. Завдяки цьому застосунки можуть обмінюватися інформацією і використовувати спільні компоненти системи.

Якщо мобільний застосунок потребує дозволів на використання конфіденційної інформації (наприклад, доступ до файлів, контактів, даних про телефонні дзвінки), список таких функцій слід також вказати у файлі конфігурації. При запуску застосунок сам зробить запит на отримання дозволу до функції ОС або даних користувача.

Мінімальна версія підтримки, поточна версія коду мобільного застосунку, версії сторонніх бібліотек вказуються у іншому файлі конфігурації з назвою «build.gradle». Цей файл містить в собі перелік налаштувань і функцій для компіляції програмного коду і збирання єдиного пакету застосунку, який можна встановити на мобільному телефоні користувача.

Особливістю безпеки програмного коду мобільних застосунків є те, що при відповідних налаштуваннях у файлі компіляції всі класи та ресурси будуть зашифровані самою ОС Android. Цю функцію було використано при написанні мобільного додатку, оскільки він є частиною системи безпеки. Таким чином, всі класи в структурі проекту, які містять в собі ключі доступу до хмарного сервісу і локальної бази даних, захищені і не можуть бути розшифровані з пакету, який є результатом компіляції всього проекту. Окрім цього, при використанні такого методу зменшується кінцевий розмір файлу компіляції, видаляється невикористаний програмний код та зайві ресурси і, як результат, оптимізується сам мобільний застосунок.

Однак в цьому методі є суттєвий мінус, який полягає у тому, що оптимізація відбувається лише для байт-коду мови програмування Java, в той же час мобільний застосунок виконується на віртуальній машині ОС Linux, яка має власний байт-код. Відповідно, оптимізація одного коду може викликати погані наслідки при його взаємодії з іншим, тому треба максимально уважно підходити до використання цієї технології і вказувати, які файли можна шифрувати, а які залишити без змін.

3.1.5 Технологія відправки повідомлень

Найважливішою функцією мобільного застосунку є вчасне інформування користувача про можливу тривогу, яку зафіксував у приміщенні мікрокомп'ютер. Існує кілька варіантів реалізаціє відправки повідомлень в режимі реального часу:

— GSM-телефонія: звичайні повідомлення, які можуть надходити через підключення мобільного телефону до стільникової мережі з використанням оператора зв'язку;

— технологія WebSocket: протокол зв'язку в мережі Інтернет, який реалізує інтерактивне з'єднання між клієнтом і сервером для обміну повідомленням в обох напрямках без затримки, в режимі реального часу. Це досить унікальна технологія, яка дозволяє без використання зайвого мережевого трафіку безпечно пересилати дані. У системах, які використовують WebSocket немає повторних запитів для підтвердження доставки, клієнт постійно тримає зв'язок із сервером і чекає від нього повідомлень. Така технологія набула свого поширення у IoT-застосунках, чатах, комп'ютерних іграх тощо;

— технологія push-повідомлень: спосіб розповсюдження інформації в мережі Інтернет, при якому дані від сервера надходять до клієнта на основі встановлених мережевих параметрів. Клієнт в свою чергу, може приймати або відхиляти повідомлення, які надходять від сервера.

Існуючі рішення систем безпеки використовують мобільну телефонію або технологію push. Технологія WebSocket є універсальним рішення для синхронізації та обміну повідомленнями, але для розгортання програми на стороні серверу необхідно здійснити дуже багато налаштувань. Це потребує багато часу, а також підключення сторонніх сервісів (хостинг, сервіси для роботи з БД тощо). Мобільний застосунок в свою чергу повинен бути завжди відкритим, інакше повідомлення не будуть надходити. Саме тому використання такої технології не є оптимальним рішенням.

Майже всі сучасні мобільні застосунки, які відповідають за отримання повідомлень або інформування користувачів використовують технологію push. Також, будь-який контент у сучасних браузерах може передаватися з використанням push-повідомлень (наприклад, реклама або інформування про появу новин). На стороні клієнта в окремому потоці постійно працює служба, яка підтримує зв'язок із сервером. Попередньо, служба має бути зареєстрована в ОС застосунку. Якщо на сервері сформовано повідомлення, воно відправляється на сторону клієнта. Кожен клієнт має власний унікальний ідентифікатор, за яким сервер знаходить необхідного користувача.

Більшість операційних систем мають свій сервер і відповідне програмне забезпечення для його розгортання. У Android — це Firebase Cloud Messaging, у iOS — APNS, у Windows Mobile — WNS. Технологія відправки push-повідомлень є досить простою в реалізації для розробника саме завдяки використання уже створених засобів для налаштування серверної частини [22]. Також великою перевагою є те, що такі повідомлення є інтерактивними для користувача, вони можуть містити в собі не тільки текст, а і графічні зображення, посилання, звукові сповіщення або навіть кнопки відповіді. В дипломному проєкті використано саме таку технологію.

Функції серверу мають прямий доступ до бази даних та хмарного сховища файлів, можуть фіксувати події додавання записів, їх видалення, оновлення. Відповідно до цього, формуються повідомлення і доставляються на мобільний телефон користувача. Push-повідомлення в собі містить інформаційний текст, а також посилання на відповідні медіафайли, які були завантажені до сховища. Дані надходять у форматі JSON і їх серіалізація здійснюється через спеціальну бібліотеку, яка підключена до проекту. Повідомлення мають високий пріоритет та звукове сповіщення і не будуть знищені або проігноровані ОС Android.

На кресленику ІА61.250БАК.005 Д3 зображено всі можливі варіанти синхронізації та обміну даних. Повідомлення до клієнта надходять на основі збережених подій у БД хмарного сервісу. Файли зберігаються окремо, і

доступ до них можливий за отриманими в повідомлення посиланнями. Відповідно, будь-який запит (наприклад, авторизація користувача в системі, отримання зображення з камери відеоспостереження в режимі реального часу, зміна налаштувань системи) відбувається через внесення змін до таблиці, яка відповідає за поточний стан системи. Дані з цієї таблиці в подальшому зчитує програма мікрокомп'ютера. Загальна схема синхронізації даних та відправки повідомлень зображена на рисунку 3.2:

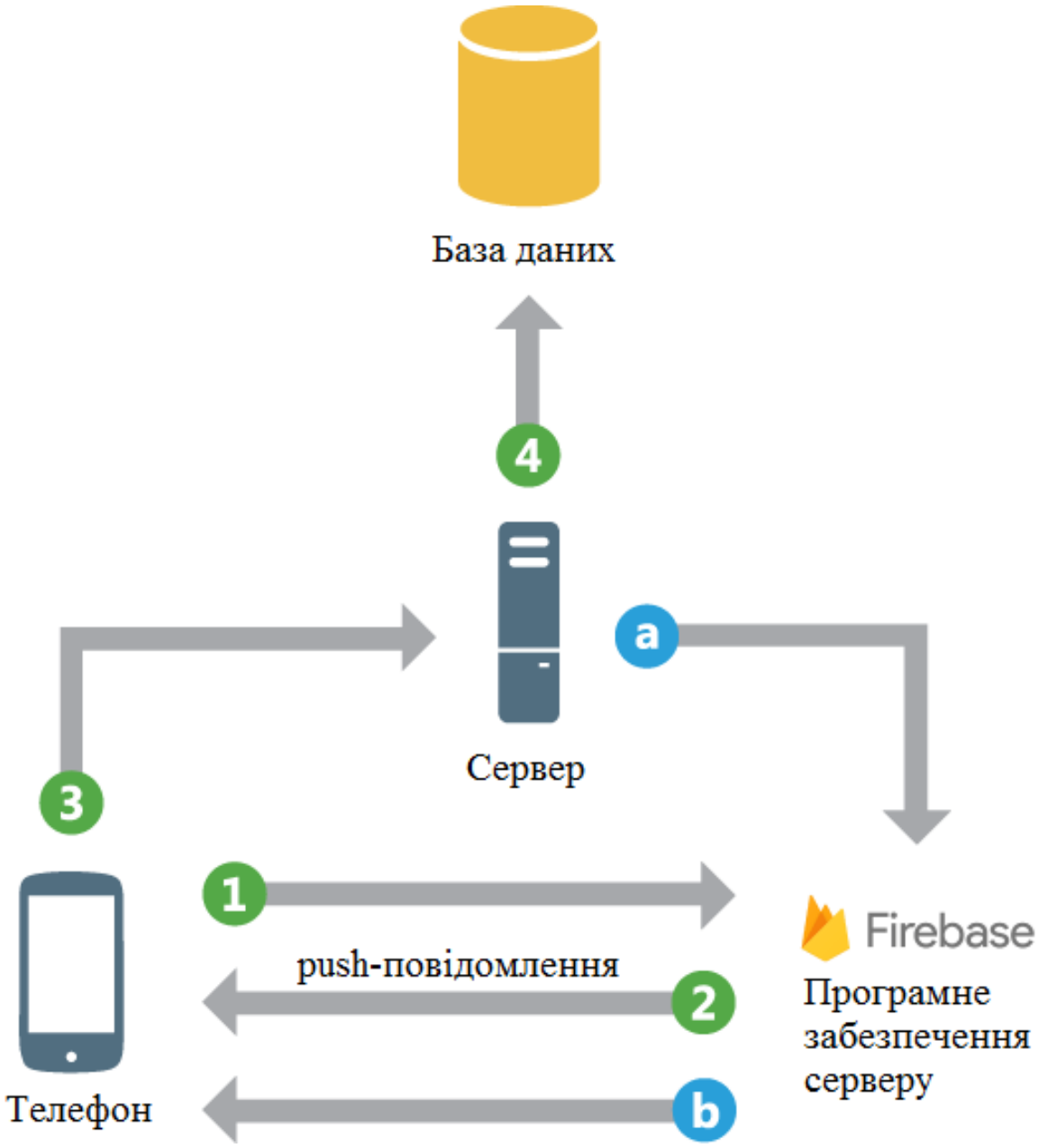


Рисунок 3.2 — Схема синхронізації даних та відправки повідомлень

3.1.6 Канали повідомлень в системі Android

Технологія комунікації між клієнтом і сервером для відправки push-повідомлень в ОС Android реалізовано через особливі служби в мобільному застосунку, які мають пріоритет в системі і не завершують свою роботу при недостатній кількості ОЗУ, що не є характерним для інших служб. Звісно, ці служби постійно несуть навантаження на ресурси мобільного телефону, однак оптимізація здійснюється на стороні сервера, який є ініціатором відправки повідомлень.

Всі повідомлення повинні бути розподілені за каналами, інакше вони не можуть відображатися на мобільному телефоні. Якщо певна категорія повідомлень є нецікавою, користувач може відмовитися від неї, але не від усіх повідомлень. Окрім цього, для кожного каналу можна налаштовувати зовнішній вигляд та звукове сповіщення повідомлень в системних налаштуваннях пристрою. Один застосунок може мати кілька інформаційних каналів (наприклад, повідомлення з різних чатів), в такому випадку розробники повинні вказувати пріоритет повідомлень (рівень їх важливості в ОС). Це необхідно для того, щоб всі повідомлення з одного каналу відображалися однаково і не знищувалися в службах ОС.

На ранніх версіях ОС Android такий розподіл за каналами відсутній, тому користувач буде отримувати абсолютно всі повідомлення і в єдиному вигляді.

При написанні мобільних застосунків, які можуть надсилати push-повідомлення, рекомендовано використовувати системний функціонал оновлення та групування повідомлень. Кожна група повідомлень може мати свою назву, зображення, за необхідності — кнопки найголовніших функцій для взаємодії з повідомленнями тощо. При надходженні нових повідомлень користувач отримує сповіщення, однак вони оновлюються всередині групи в хронологічному порядку. Завдяки такому підходу інформаційні повідомлення з одного застосунку будуть відображатися єдиною групою і користувач може

розкривати всю групу з метою пошуку необхідних повідомлень. Це оптимізує графічний інтерфейс і зменшує навантаження на користувача. Приклад групування повідомлень з одного мобільного застосунку зображено на рисунку 3.3:

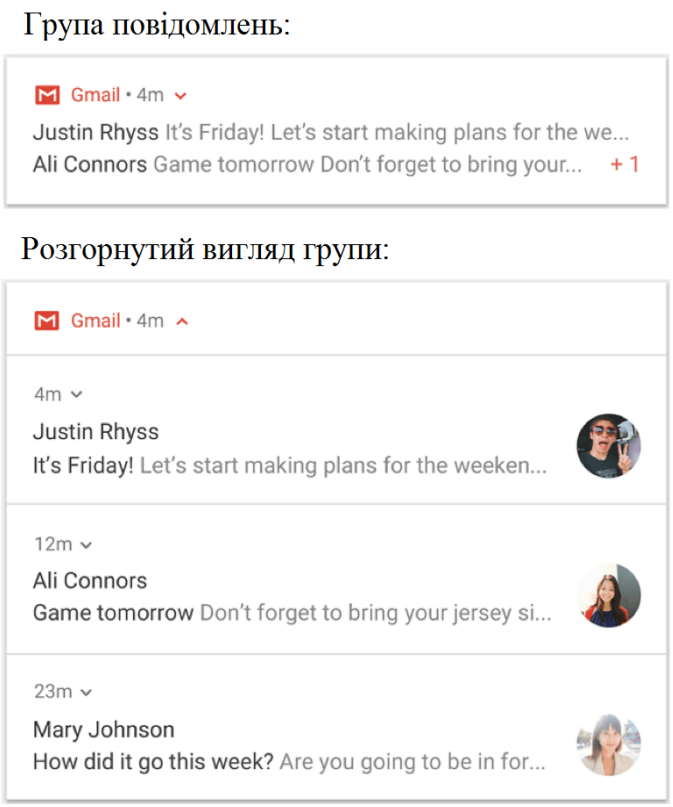


Рисунок 3.3 — Приклад групування push-повідомлень з одного мобільного застосунку

На мобільних пристроях з ОС Android версії 5.0 та вище є можливість налаштування режиму, в якому відключається звукові сповіщення та вібросигнал. При цьому, повідомлення відображається в графічному інтерфейсі, поки користувач не внесе відповідні зміни до налаштувань каналів.

3.1.7 Графічний інтерфейс

Мобільний додаток є інтерактивною програмою з елементами графічного інтерфейсу. Користувач керує системою безпосередньо через

команди, які надходять від його взаємодії з екранами графічного інтерфейсу. Кожен екран має власну модель представлення даних (компонент патерну MVVM з назвою «ViewModel»). Модель передає дані від серверу до елементів графічного інтерфейсу, і навпаки. Зв'язок реалізовано через використання технології «databinding», яка дозволяє конкретним змінним або методам у програмному коді змінювати графічні елементи представлення (в ОС Android технологія реалізована на базі засобів реактивного програмування віртуальної машини Java).

Більшість екранів містять в собі стандартні UI-компоненти операційної системи Android. Це текстові і графічні зображення, кнопки, елементи вибору, діалоги, поля для вводу текстових даних тощо. Компоненти групуються у файлах розмітки інтерфейсу, які знаходяться в окремих директоріях серед ресурсів проекту. Також необхідною складовою реалізації графічного інтерфейсу користувача є графи навігації між екранами, які забезпечують інтуїтивність та зручність користування програмою. Окрім цього, графи навігації слугують для передачі параметрів (аргументів) від одного екрану до іншого.

В структурі проєкту є окремі файли з текстовими ресурсами, кольорами інтерфейсу, стилями мобільного застосунку, а також анімації переходу між екранами. Доступ до ресурсів з програмного коду можливий через ініціалізацію відповідних змінних, які тримають у собі посилання на різні директорії і файли в проєкті. Екрани графічного інтерфейсу описуються мовою розмітки «xml». Також варто сказати, що ця мова використовується для серіалізації моделей і передачі даних. Графічні ресурси можуть бути оптимізовані під різні версії ОС Android та екрани мобільних пристроїв, в такому випадку до структури проєкту додаються директорії з посиланням на ресурси для використання на тому чи іншому пристрої.

Варто сказати, що програмний код не залежить від файлів розмітки графічного інтерфейсу. При ініціалізації того чи іншого екрану підтягується відповідний файл розмітки, але основний функціонал, який прописаний в

класах, може бути використаний навіть без графічного інтерфейсу. При компіляції мобільного застосунку утворюється «.apk»-файл, в якому окремо знаходиться байт-код програми та всі ресурси, стилі, графічні зображення, анімації тощо.

При відкритті мобільного застосунку користувачу буде запропоновано пройти авторизацію через існуючий в системі профіль або створити новий. В майбутньому, користувач може здійснити вихід із свого облікового запису на екрані налаштувань. В такому випадку, для отримання повторного доступу до функціоналу системи мобільний застосунок запропонує заново авторизуватися. Це досить важливий момент з точки зору безпеки даних в системі. Мобільний застосунок визначає авторизований стан користувача через збережений в локальній базі даних ідентифікатор доступу (токен, необхідний для здійснення запитів до хмарного сховища медіафайлів та БД на сервері). При виході з профілю, всі локальні налаштування та інформація про користувача видаляється в мобільному застосунку. Дані зберігаються лише на сервері для повторного входу в систему. Графічний інтерфейс екранів авторизації («Вхід» та «Реєстрація») зображено на рисунку 3.4:

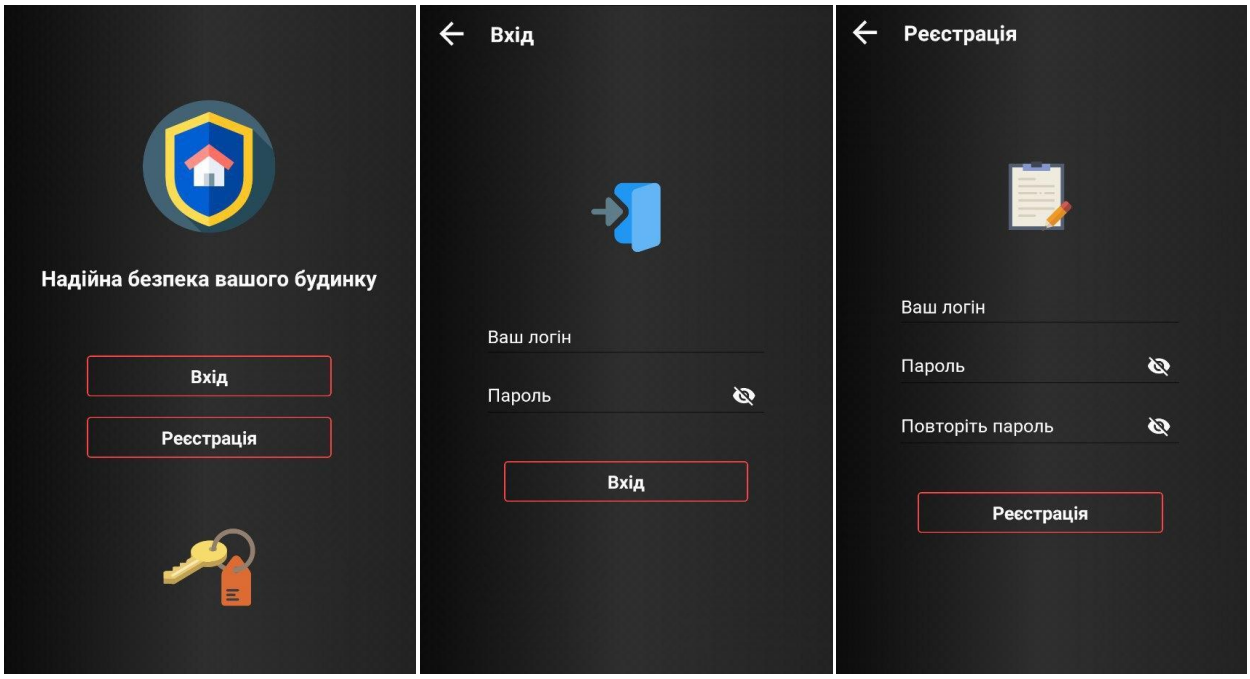


Рисунок 3.4 — Інтерфейс екранів авторизації

Після успішної авторизації або створення нового прфілю користувачу доступний основний функціонал програми. На головному екрані є можливість дистанційно відключити охоронну систему або активувати її через відповідний елемент графічного інтерфейсу. Також звідси можлива навігація до інших екранів застосунку. В залежності від обраного стану системи, головний екран може змінювати своє представлення (рисунок 3.5).

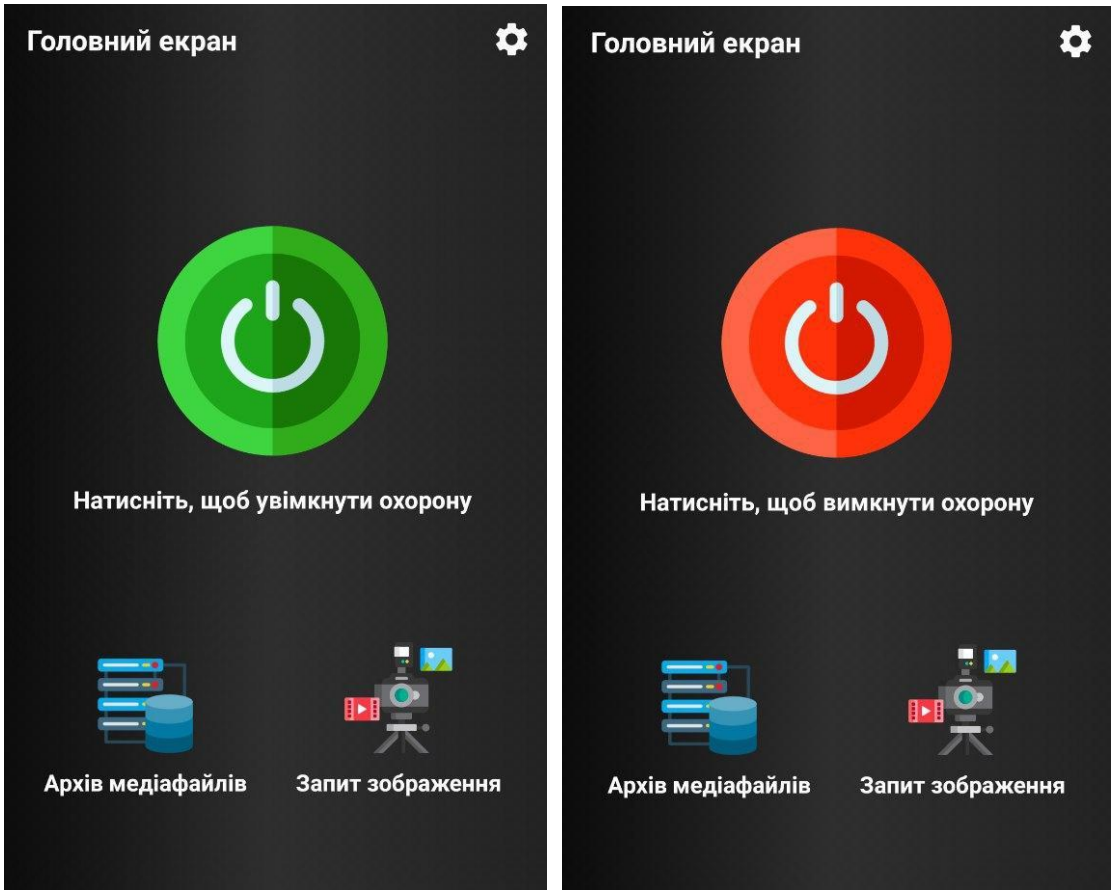


Рисунок 3.5 — Головний екран застосунку при різних станах системи

Також на цьому екрані є можливість відкрити налаштування застосунку, перейти до архіву медіафайлів або зробити запит на отримання зображення з відеокамери в режимі реального часу через відповідне діалогове вікно (рисунок 3.6). Відносно останньої функції, система спочатку запитує користувача, який тип медіафайлу він бажає отримати, серед можливих варіантів — фото та відео. Даний функціонал працює навіть якщо система відключена в поточний момент і мікрокомп'ютер знаходиться в режимі очікування. Реалізовано це через синхронізацію інформації в базі даних, до

якої підключена програма користувача з одного боку, та мікрокомп'ютер — з іншого. До бази даних вноситься відповідний запис на отримання зображення, і на це миттєво реагує програма мікрокомп'ютера, яка виконується в циклічному режимі. Звісно, тут є певна затримка (до 10 секунд), оскільки потрібен час на синхронізацію та завантаження файлів до хмарного сховища перед тим, як вони будуть відправлені на телефон.

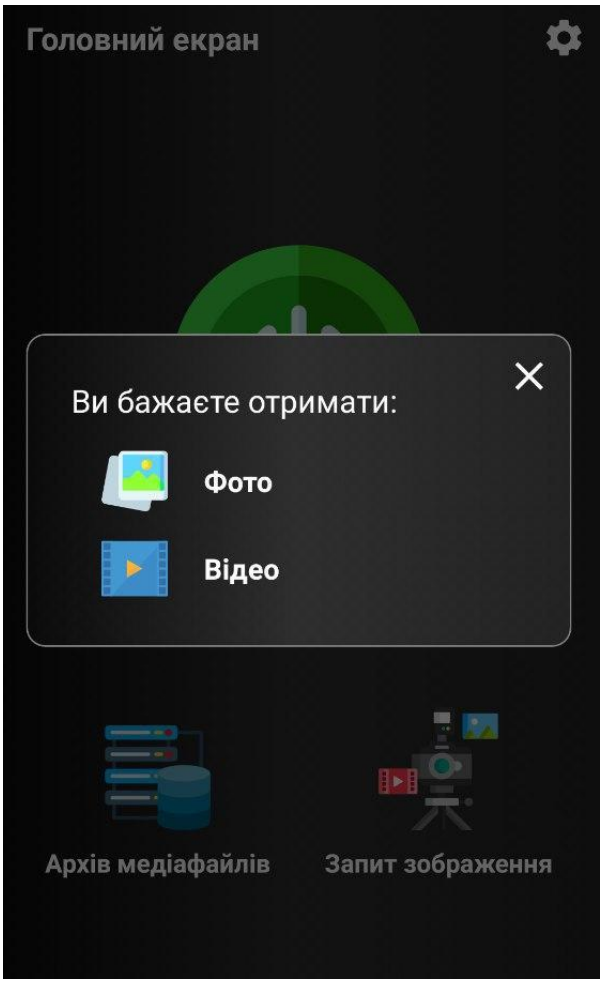


Рисунок 3.6 — Запит на отримання зображення з відеокамери

Якщо в приміщенні датчиком зафіксовано рух, користувач мобільного застосунку миттєво отримує повідомлення, яке можна переглянути серед списку отриманих сповіщень на мобільному телефоні (рисунок 3.7). Окрім даного списку, повідомлення буде відображатися на заблокованому екрані телефону, а також на підключених пристроях-носіях. Повідомлення не може бути проігнороване ОС і залишається доступним для перегляду до того

моменту, поки його не відкриє користувач. В програмному коді служби відправки повідомлень встановлено найвищий пріоритет для мобільного застосунку, тому телефон інформує користувача відповідним звуковим сповіщенням та вібросигналом [23].

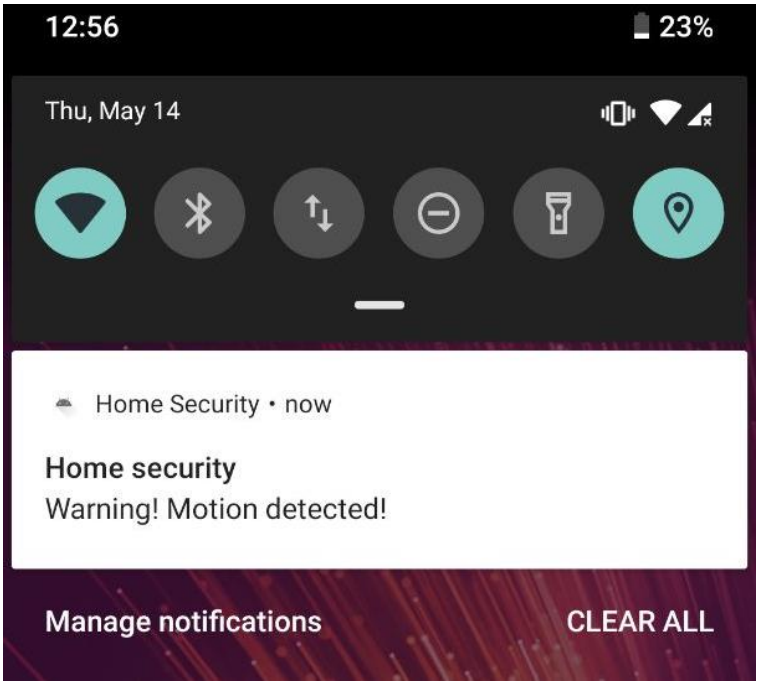


Рисунок 3.7 — Push-повідомлення на мобільному телефоні

Якщо користувач натисне на повідомлення, ОС Android автоматично відкриє мобільний застосунок і користувач відразу потрапляє на екран перегляду медіафайлу. Це реалізовано через системні засоби прив'язки повідомлень до мобільних застосунків, які їх генерують. При відправці повідомлення з програмного коду потрібно лише вказати активність, яка має бути запущена в ОС після відкриття повідомлення. Інакше кажучи, це екран графічного інтерфейсу або служба, які можуть бути викликані через повідомлення.

Після цього здійснюється завантаження медіфайлу за посиланням, яке надійшло у повідомленні. Звісно, через це є певна затримка, але такий підхід набагато ефективніший, оскільки push-повідомлення займає дуже малий об'єм мережевого трафіку і надходить миттєво від хмарного сервісу на сторону клієнта. Якщо файл успішно завантажено, його можна переглянути (це може

бути фото або від в залежності від встановлених користувачем налаштувань). Графічний інтерфейс екрану, який відповідає за перегляд і завантаження медіафайлів, зображено на рисунку 3.8:

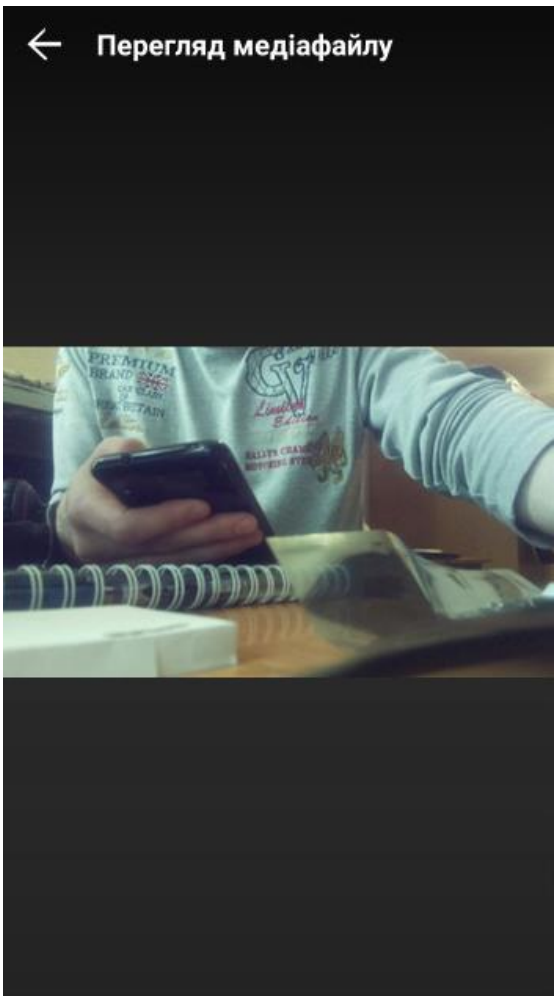


Рисунок 3.8 — Екран перегляду зображення, яке було отримано з камери відеоспостереження

Всі медіафайли зберігаються і можуть бути доступні для наступних переглядів на окремому екрані застосунку «Архів медіафайлів». Фото та відео зберігаються за різними посилання в хмарному середовищі, тому для зручності окремо можна переглянути всі файли того чи іншого типу. Якщо у користувача буде необхідність видалити певний файл з архіву, такий функціонал також є в мобільному застосунку.

Для зручного перегляду, фото відображаються безпосередньо на екрані зі списком і можуть бути відкриті для уточнення деталей в окремому вікні

перегляду. В свою чергу, для відеозаписів така можливість не передбачена і перегляд доступний після вибору необхідного файлу лише на окремому екрані з фунціоналом завантаження і відтворення відео в системному програвачі ОС Android. Графічний інтерфейс екранів роботи з архівом медіафайлів в мобільному застосунку зображено на рисунку 3.9:

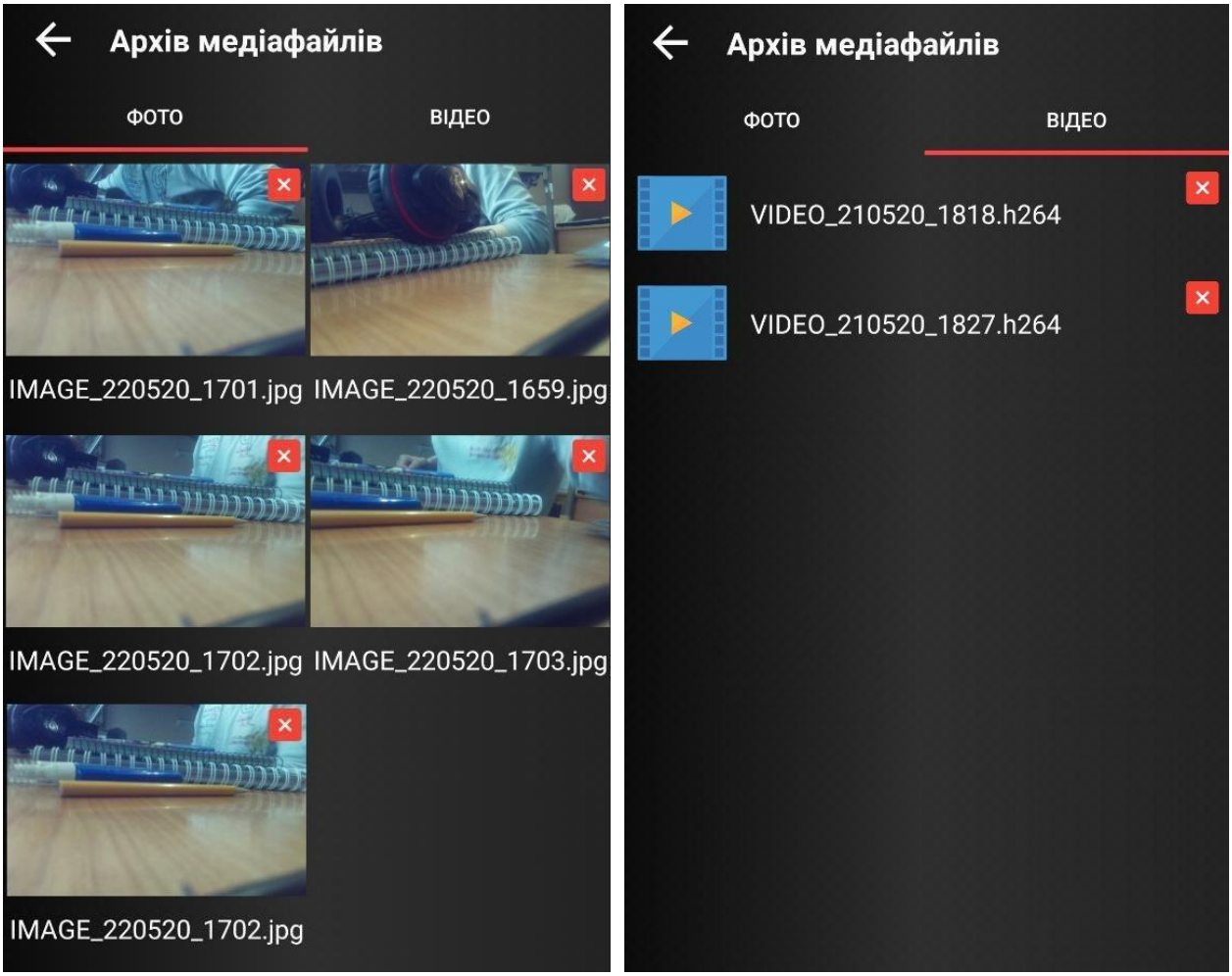


Рисунок 3.9 — Графічний інтерфейс архіву медіафайлів в мобільному застосунку (фото та відео)

Користувач може змінити тип медіафайлу, який система буде надсилати у випадку тривоги і якщо обрано відео, додатково можна встановити тривалість запису (5, 7 або 10 секунд). Розмір файлу прямо пропорційно залежить від цього показника, тому при збільшенні тривалості запису збільшується і затримка сповіщення, оскільки система потребує більше часу на завантаження файлів до хмарного сховища.

Керувати налаштуваннями системи можна через окремий екран графічного інтерфейсу в мобільному додатку (рисунок 3.10):

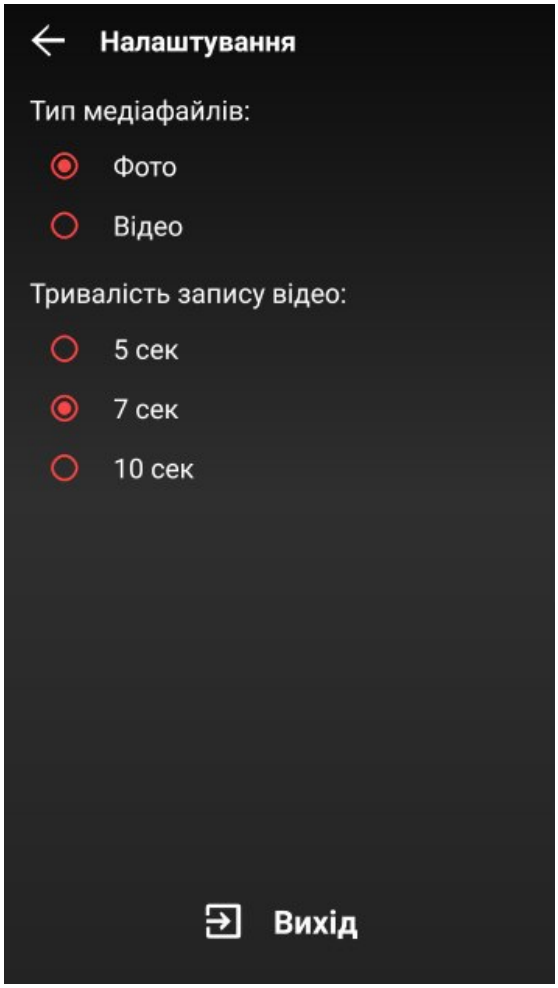


Рисунок 3.10 — Функціонал екрану налаштувань

3.1.8 Навігація між екранами

При наявності великої кількості екранів графічного інтерфейсу, з якими може взаємодіяти користувач, виникає необхідність створення відповідної навігації для зручного користування програмою та функціоналом. Для вирішення цієї проблеми існує окремий архітектурний компонент — граф навігації. Після створення всі екранів інтерфейсу окремо вони можуть поєднуватися зв'язками у графах, які містять в собі всю інформацію про переходи між екранами, передачу параметрів, повернення назад до раніше

відкритих екранів тощо. Графи можуть бути багаторівневими, тобто включати в себе менші за кількістю екранів компоненти навігації. Такий підхід з використанням декомпозиції надає можливість легко тестувати програму та знаходити помилки в окремих модулях навігації [24].

Створений мобільний застосунок містить 2 активності: AuthActivity (авторизація) та MainActivity (основний функціонал). Відповідно, в ресурсах програми створено 2 графи навігації. Всі екрани описуються тегами у відповідних файлах типу «.xml».

На рисунку 3.11 зображено навігацію між екранами авторизації. Вона містить всього лише 3 екрани, як відповідають за функціонал входу з використанням існуючого профілю або реєстрація користувача в системі.

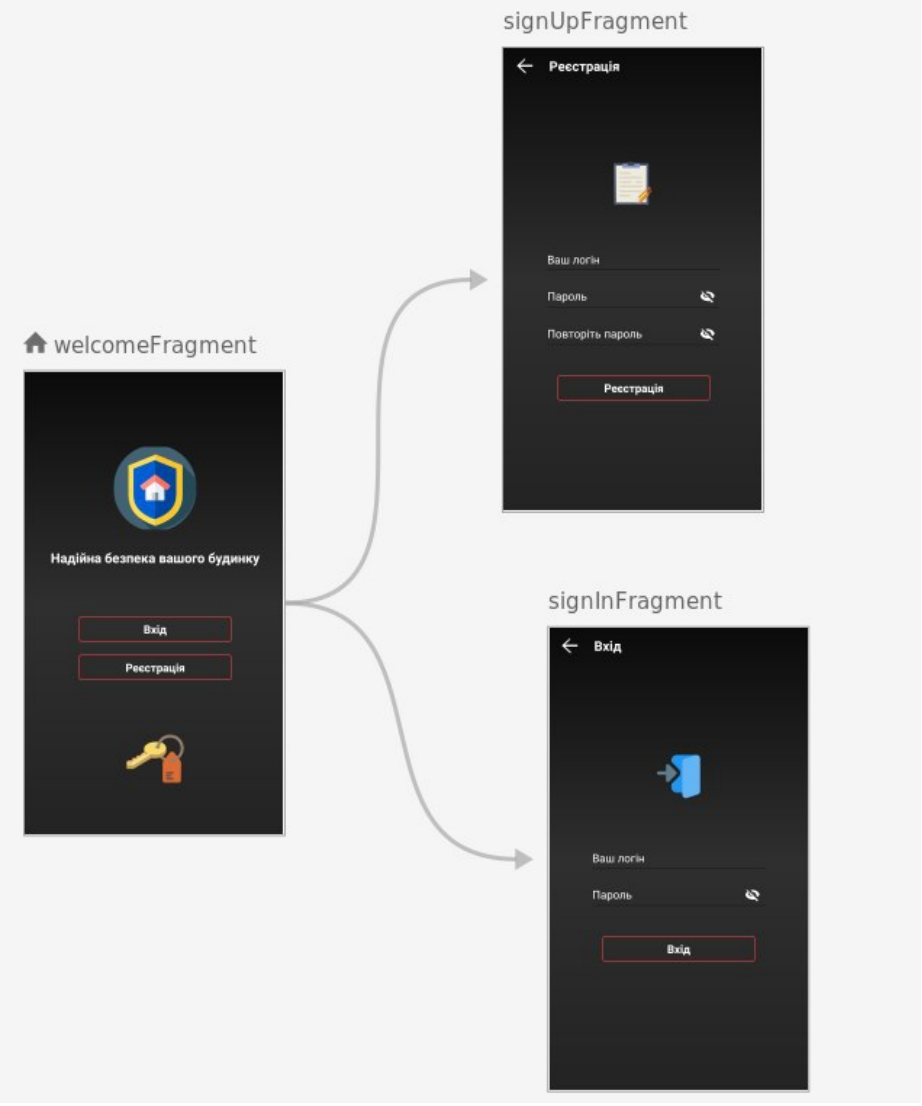


Рисунок 3.11 — Навігація між екранами авторизації

За навігацію між екранами основного функціоналу мобільного застосунку відповідає граф, зображений на рисунку 3.12. Він містить в собі головний екран меню, архів медіафайлів, налаштування та екран перегляду і завантаження медіафайлу, який був отриманий при інформуванні користувача.

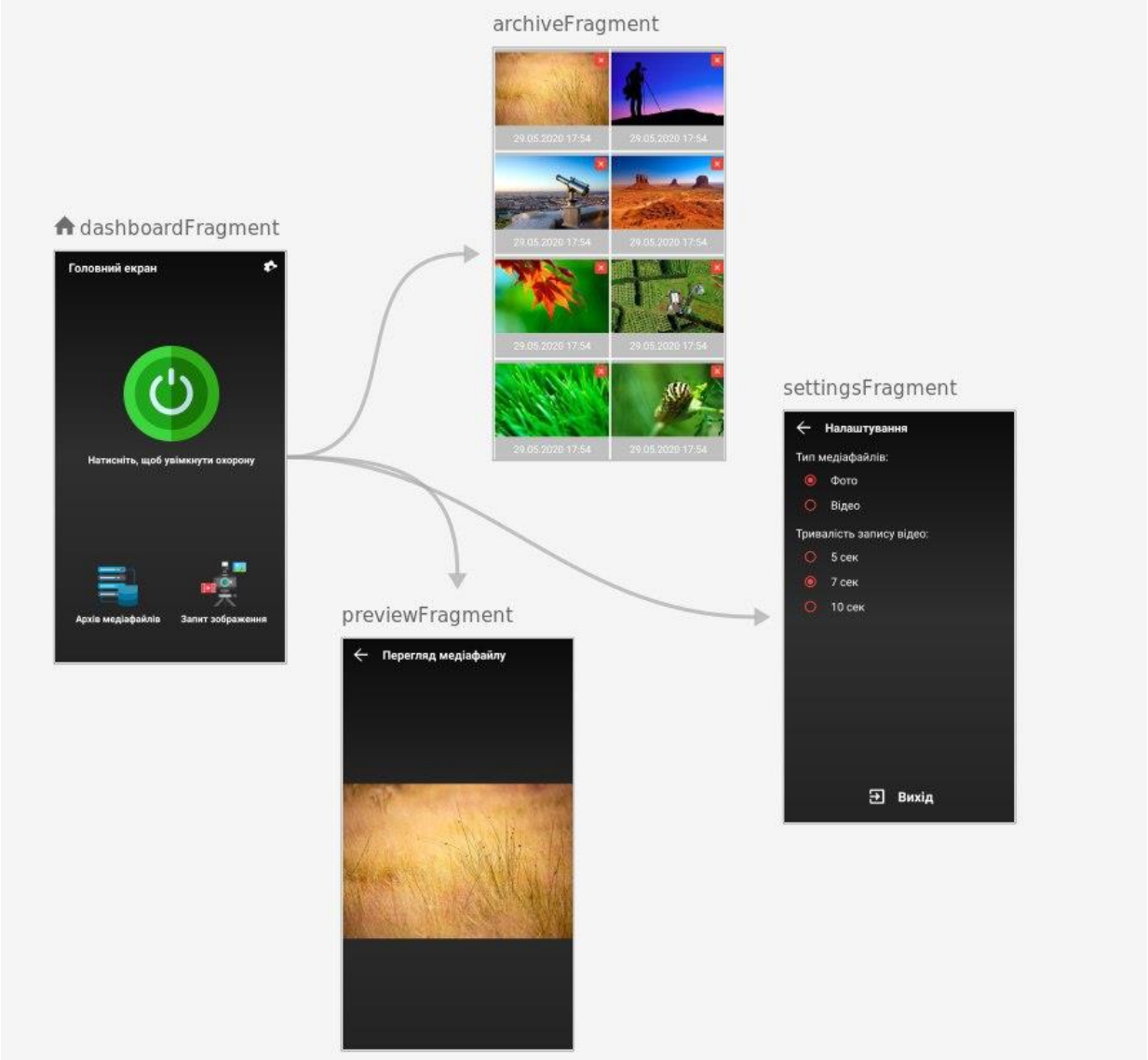


Рисунок 3.12 — Навігація між екранами основного функціоналу мобільного застосунку

Графи навігації дають можливість контролювати всі екрани мобільного застосунку, оптимізувати певні сценарії для зручного користування, передавати аргументи між екранами тощо.

3.2 Розробка та опис програми мікрокомп'ютера

Для розділення функцій і структуризації програми окремо винесені методи авторизації, генерації імені файлів, збереження їх в файловій системі та завантаження на сервер. Головна частина включає програмний код моніторингу стану системи, зчитування налаштувань і запису інформації про події до бази даних (лістинг коду — додаток Б).

3.2.1 Середовище розробки

Програма мікрокомп'ютера написана мовою Python з використанням середовища Thonny Python IDE, яке встановлюється в якості рекомендованого продукту безпосередньо виробником Rasberry Pi і є досить популярним та простим інструментом для програмування всіх поколінь мікрокомп'ютерів сімейства Rasberry. Це середовище розробки дозволяє підключати різні пакети для керування мікрокомп'ютером і периферійними пристроями, а також підключати потрібні сервіси в якості сторонніх бібліотек і легко їх використовувати в програмному коді. Зокрема, написана в дипломному проєкті програма працює з виходами загального призначення (GPIO) на платі мікрокомп'ютера, камерою, файловою системою ОС, а також інтегрується з хмарним сервісом Firebase.

3.2.2 Короткий опис використаних бібліотек

Програма мікрокомп'ютера використовує кілька системних бібліотек АП Rasberry Pi. Серед них є бібліотека «RPi.GPIO», яка відповідає за взаємодію між алгоритмом програми та виходами загального призначення на платі мікрокомп'ютера. Датчик руху має один інформаційний канал, який надсилає дані до системи саме через вихід GPIO. Попередньо він ініціалізується в програмному коді та відповідає значенню однієї змінною, яка може керувати станом системи. Бібліотеки «Datetime» і «Time» необхідні для

отримання актуального системного часу та реалізації затримки виконання програмного коду, оскільки він виконується в циклічному режимі.

Для роботи з камерою підключена бібліотека «PiCamera». Саме через цей пакет системних засобів здійснюється підключення модулю відеокамери, а також встановлюються базові налаштування (розширення знімків, тривалість запису відео, положення камери у просторі тощо).

Окрім системних бібліотек, до програмного коду підключена одна стороння бібліотека — «Pyrebase». Це безкоштовний продукт для спрощення роботи із хмарним сервісом Fyrebase, саме з використанням мови Python. Містить в собі безліч методів для авторизації, роботи з базою даних (запис, зчитування, оновлення інформації в таблицях) та сховищем файлів. Програмний код здійснює авторизацію мікрокомп'ютера, синхронізацію налаштувань та завантажує файли саме з використанням цієї бібліотеки.

3.2.3 Алгоритм роботи програми

Написаний програмний код виконується в циклічному режимі, а завантаження програми на виконання здійснюється при старті операційної системи, тобто при підключенні живлення до мікрокомп'ютера. На самому початку виконання програми здійснюється авторизація та підключення до сервісу Firebase, з якого завантажуються налаштування користувача, стан системи (увімкнена чи відключена користувачем в ручному режимі). Після авторизації програма отримує доступ до бази даних та хмарного сховища файлів, куди може завантажувати фото та відео з камери для перегляду на стороні мобільного застосунку. Окрім цього, перед початком роботи в циклічному режимі, програма налаштовує відеокамеру (розширення для знімків та запису відео), встановлює режим зчитування даних від датчика руху.

Алгоритм роботи головної частини програмного коду наступний. З хмарного сервісу програма зчитує налаштування, які вставлені користувачем

через мобільний застосунок. При запиті на отримання фото чи відео в режимі реального часу ці налаштування змінюються і програма змінює алгоритм відповідно до вимог користувача. В звичайному режимі здійснюється моніторинг руху в приміщенні через датчик обходу перешкод, який може фіксувати наявність переміщень предметів в полі його зору. Здійснюється таке спостереження за станом приміщення в постійному циклі, з інтервалом в 2 секунди. Якщо датчик фіксує рух, програма вмикає відеокамеру і, в залежності від налаштувань користувача, робить знімок або записує відео, тривалістю до 10 секунд. Ці файли відразу зберігаються у відповідній директорії файлової системі. Назви всіх файлів формуються на основі вибраного типу (фото чи відео) та поточного часу події, яку зафіксував мікрокомп'ютер. Це дозволяє потім переглядати на мобільному телефоні історію створення файлів в хронологічному порядку. Разом з локальним шляхом в файлові системі мікрокомп'ютера одночасно формується і посилання на хмарне сховище, за яким буде завантажено файл. Відповідно, після цього програма здійснює завантаження файлу за створеним посиланням і робить запис до бази даних про те, що було зафіксовано нову подію. Цей запис в подальшому є ініціатором відправки повідомлення користувачу на мобільний телефон.

Якщо користувач вимкнув охорону, то в циклічному режимі весь описаний вище функціонал не виконується і програма перебуває в режимі очікування зміни стану, при цьому постійно зчитує лише налаштування з бази даних. Аналогічною є поведінка в ситуації, коли датчик не фіксує руху в приміщенні. В таких випадках вся система перебуває в режимі очікування і на початку виконання кожного циклу фіксує зміни або їх відсутність. Блок-схема алгоритму роботи програми мікрокомп'ютера наведена у додатку А.

Варто звернути увагу ще на один проміжний етап, коли користувач через мобільний застосунок робить запит на отримання зображення або відео в режимі реального часу. Це здійснюється через внесення відповідних змін до бази даних з боку клієнта, ці зміни зчитує програма мікрокомп'ютера і після

відправки файлів повертає відповідні значення налаштувань в базі даних до початкових. Після цього користувач отримує результат запиту і може переглянути фото або відео на мобільному телефоні, а програма мікрокомп'ютера повертається в звичайний режим моніторингу.

3.3 Розробка та опис хмарного сервісу

По аналогії використання функцій виділеного фізичного серверу в архітектурі системи, на базі можливостей Firebase створено функції синхронізації, управліннями даними і відправки повідомлень, які працюють постійно і не залежать від підключених клієнтів. Створені вони з використанням програмної платформи «Node.js» [25]. Наприклад, можна описати найголовнішу функцію, яка є ініціатором відправки повідомлення на мобільний телефон користувача саме в той момент, коли мікрокомп'ютер зафіксував рух і зробив відповідний запис до бази даних. Лістинг коду функції відправки повідомлень наведено у додатку В.

В даному випадку можна сказати, що хмарний сервіс виступає посередником для комунікації між клієнтами абсолютно різних платформ.

Всі файли (фото та відео) зберігаються в хмарному сховищі у відповідних директоріях, таких як «image/» та «video/». Доступ до цих файлів мають лише авторизовані користувачі мобільного застосунку, а також програма мікрокомп'ютера, яка і завантажує файли до сховища.

Оскільки система забезпечує охорону, в ній обов'язково має бути лише авторизований доступ. На стороні хмарного сервісу, для кожної сесії мобільного застосунку генерується унікальний хеш, який зберігається для відправки повідомлень тільки на потрібний, авторизований пристрій користувача. Програма мікрокомп'ютера, в свою чергу, містить в собі також унікальні ідентифікатори доступу, які дозволяють їй зчитувати налаштування користувача з бази даних і зберігати файли в сховищі.

4 ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ СТВОРЕНОЇ СИСТЕМИ

Особливістю створеної в дипломному проєкті системи є інтеграція 3 різних платформ, серед яких Android (клієнт), Rasberry Pi (виконавчий мікрокомп'ютер) та сервіс Firebase (хмарна технологія для синхронізації та збереження даних). Відповідно, існують 3 модулі програмного коду, які не залежать один від одного і кожен з яких виконується тільки на одній платформі.

Мобільний застосунок можна встановлювати на будь-які телефони з операційною системою Android версії 5.0 та вище. Мінімальна версія підтримки вказується в програмному коді компіляції застосунку і не може змінюватися. На мобільному телефоні повинен обов'язково бути доступ до мережі Інтернет. Цього потребує і сам мікрокомп'ютер для функціонування системи і синхронізації даних. Можливі варіанти підключення — локальна мережа Wi-Fi в приміщенні або безпосередньо з використанням кабелю Ethernet, який можна під'єднати до відповідного порту. Також необхідністю для роботи мікрокомп'ютера є наявність живлення від електромережі з напругою 220В.

При використанні даної системи або аналогічних слід враховувати оптимальні умови використання (освітлення, температурний режим тощо). Мікрокомп'ютер при значному навантаженні може нагріватися до незвичних для його роботи температур, тому варто додатково встановлювати будь-яку елементарну систему охолодження. Звісно, якщо програма не навантажує платформу, цього можна уникнути. При використанні різних датчиків руху (особливо, якщо їх основним елементом є інфрачервоний світловий діод) слід подумати про освітлення у приміщенні, де такі датчики будуть встановлені. Наприклад, використаний в дипломному проєкті інфрачервоний датчик реагує на будь-які переміщення в просторі, проте має певні обмеження, а саме: нестабільність роботи при поганому освітленні і максимальна відстань фіксації руху становить 2 м. Даний показник відстані прямо пропорційно

залежить від рівня освітлення, тому для корегування чутливості на самому датчику є потенціометри.

Функціонал системи можна легко розширити. Наприклад, додати нові типи датчиків або звукове сповіщення в будинку при фіксації руху. Такі функції досить легко реалізувати на обраній для дипломного проєкту моделі мікрокомп'ютера. На ньому є велика кількість різних виходів для управління виконавчими пристроями або для створення додаткових інформаційних каналів від встановлених датчиків. Варто сказати, що кожне наступне покоління АП Rasberry Pi надає розробникам все більше і більше можливостей, особливою перевагою покоління Model 3B+ є використання Ethernet технології для розгортання локальних мереж.

Аналогічно можна розширити і список доступних функцій мобільного застосунку. Як варіант, зробити локальне сховище медіафайлів, створити додатковий інформаційний канал на основі GSM-телефонії (це відразу підвищує надійність системи і така реалізація є досить простою з використанням ОС Android). Але навіть через один канал зв'язку (мережа Інтернет) створена система не поступається аналогам по надійності інформування користувачів, оскільки використано один із найсучасніших сервісів синхронізації та відправки повідомлень.

ВИСНОВКИ

Проблема безпеки сучасних приватних будинків або навіть житлових комплексів є особливо актуальною на сьогоднішній день. Створення автономних систем, в тому числі і охоронних, стало значно простішим, ніж у минулому. В першу чергу, це пов'язано з розвитком мікропроцесорної техніки, сучасних телекомунікаційних мереж та інформаційних технологій.

Після детального аналізу предметної області та існуючих рішень був сформований перелік основного функціоналу системи безпеки, а також виділено ряд фізичний пристроїв, необхідних для реалізації системи. Виконано всі вимоги поставленої задачі дипломного проєктування, а саме:

— реалізовано методи комунікації між апаратними платформами Rasberry Pi та Android;

— створено мобільний застосунок користувача, написано програму мікрокомп'ютера та функції синхронізації даних на сервері;

— розроблено реальний прототип системи безпеки, яка складається з мікрокомп'ютера, периферійних пристроїв і відповідного програмного забезпечення. Прототип системи повністю робочий і готовий для використання, а існуючий функціонал може бути легко розширений шляхом додавання, наприклад, датчиків різного типу, резервних каналів зв'язку, системи звукової тривоги в приміщенні тощо.

Система описана з використанням схеми підключення пристроїв мікрокомп'ютера, структурної схеми системи, діаграми потоків даних та діаграми класів мобільного застосунку. В пояснювальній записці детально описано використану технологію обміну даними між апаратними платформами, функціонал і основні архітектурні компоненти мобільного додатку, алгоритм роботи програми мікрокомп'ютера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

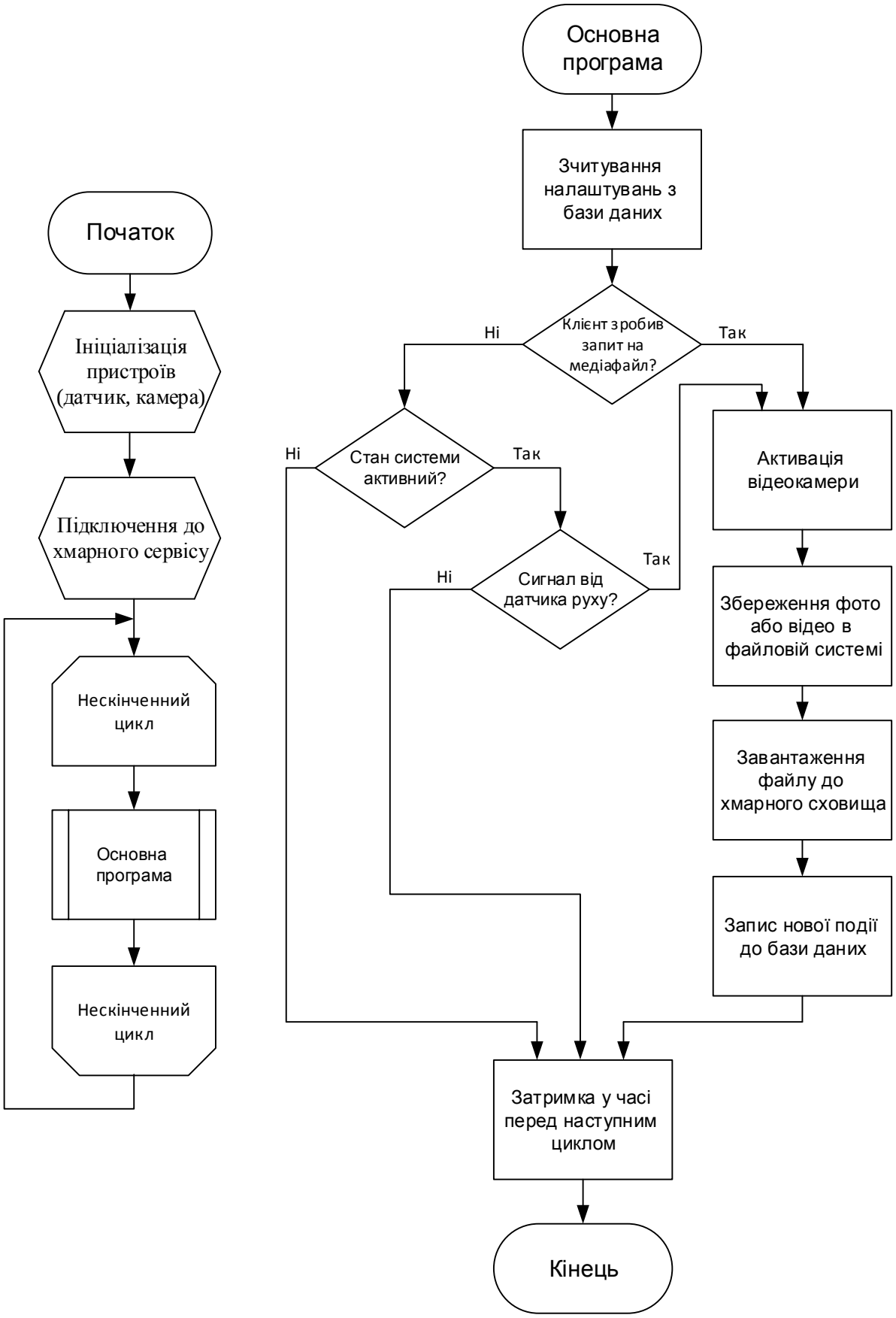
1. Зображення систем «розумного будинку» [Електронний ресурс]: Режим доступу: https://melask.com.ua/smart_house
2. Procedia Computer Science. Smart Home: Architecture, Technologies and Systems [Електронний ресурс]: Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877050918305994 - 11.05.2018 р.
3. Home security system using internet of things [Електронний ресурс] : Режим доступу: https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042026/pdf - 11.05.2018 р.
4. Зображення охоронної системи «Ajax Starter Kit Cam» [Електронний ресурс]: Режим доступу: https://alarm.bezpeka.systems/product/ajax-starterkit-cam-black/
5. Зображення охоронної системи «GSM 10GA» [Електронний ресурс]: Режим доступу: https://adviser.in.ua/komplekt-besprovodnoy-ulichnoy-gsm-signalizacii-gsm-10ga-ik-barery.html
6. Pros and Cons of Cloud Storage [Електронний ресурс]: Режим доступу: https://www.securestorageservices.co.uk/article/11/pros-and-cons-of-cloud-storage - 04.07.2018 р.
7. Firebase, єдиний крос-платформний SDK від Google [Електронний ресурс]: Режим доступу: https://developer.android.com/distribute/best-practices/develop/build-with-firebase - 27.12.2019 р.
8. OAuth 2.0 Client Credentials Grant [Електронний ресурс]: Режим доступу: https://oauth.net/2/grant-types/client-credentials/ - 13.06.2012 р.
9. Зображення мікрокомп'ютера Rasberry Pi Model 3B+ [Електронний ресурс]: Режим доступу: https://www.distrelec.biz/en/raspberry-pi-model-1gb-ram-raspberry-pi-raspberry-pi-3b/p/30109158
10. Виктор Петин, Arduino и Rasberry Pi в проектах Internet of Things [Текст] - Санкт-Петербург: издат. «БВХ-Петербург», 2019. – 432 с.

11. С. Могильний, Мікрокомп'ютер Rasberry Pi - інструмент дослідника. Рішення апаратних задач [Текст] - К.: видавн. «Талком», 2014. - 340 с.
12. Зображення портів загального призначення GPIO на платі мікрокомп'ютера Rasberry Pi [Електронний ресурс]: Режим доступу: https://miniboard.com.ua/boards/854-raspberry-pi-4-model-b.html
13. Зображення нумерації портів GPIO [Електронний ресурс]: Режим доступу: https://pinout.xyz/
14. Порты GPIO в Rasberry Pi [Електронний ресурс]: Режим доступу: https://ph0en1x.net/86-raspberry-pi-znakomstvo-s-gpio-perekluchatel-i-svetodiod.html#what-is-gpio - 02.08.2016 р.
15. Getting started with the Camera Module [Електронний ресурс]: Режим доступу: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera - 29.02.2019 р.
16. Зображення модулю камери Rasberry Pi [Електронний ресурс]: Режим доступу: https://robot-on.ru/articles/raspberry-pi-rabota-s-kameroi
17. Зображення інфрачервоного датчика руху [Електронний ресурс]: Режим доступу: https://robotdyn.com/infrared-barrier-sensor-obstacle-avoidance.html
18. Ян Ф. Дарвин, Android. Сборник рецептов: задачи и решения для разработчиков приложений [Текст] - К.: издат. «Диалектика», 2018. - 768 с.
19. Створення мобільних застосунків під ОС Android. Основні принципи та базові компоненти системи [Електронний ресурс]: Режим доступу: https://developer.android.com/guide/components/fundamentals -27.12.2019 р.
20. Фонові служби в ОС Android [Електронний ресурс]: Режим доступу: https://developer.android.com/guide/components/services - 27.12.2019 р.
21. Определение паттерна MVVM [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/wpf/22.1.php - 18.10.2016 р.
22. Firebase Cloud Messaging [Електронний ресурс]: Режим доступу: https://firebase.google.com/docs/cloud-messaging/ - 06.05.2020 р.
23. Огляд повідомлень [Електронний ресурс]: Режим доступу: https://developer.android.com/guide/topics/ui/notifiers/notifications

24. Principles of navigation [Електронний ресурс]: Режим доступу: https://developer.android.com/guide/navigation/navigation-principles - 27.12.2019 р.
25. Cloud Functions For Firebase [Електронний ресурс]: Режим доступу: https://firebase.google.com/docs/functions/ - 13.04.2020 р.
26. Оформлення текстових документів у навчальному процесі. Стандарт організації (кафедри) СОУ АУТС 01-15. Для студентів кафедри автоматики та управління в технічних системах [Текст] / Уклад.: Я.Ю. Дорогий, Н.Б. Репнікова, О.І. Ролік, Л.Ю. Юрчук – К.: НТУУ «КПІ», 2015. – 27 с.
27. ДСТУ 1.5 : 2003 «Правила побудови, викладання, оформлення та вимоги до змісту нормативних документів». «Державна система стандартизації України. Загальні вимоги до побудови, викладу, оформлення та змісту стандартів» [Текст]. – На заміну ДСТУ 1.5-93 ; Чинний від 2003-07-01. – Київ : Видавництво стандартів, 2003. – 44 с.
28. Дипломний проєкт бакалавра. Розробка, оформлення, захист [Електронний ресурс]: навч. посіб. для студ. які навчаються за напрямками підготовки 151 Автоматизація та комп'ютерно-інтегровані технології та 121 Інженерія програмного забезпечення / Уклад.: Я.Ю. Дорогий, К.С. Дорошенко, Н. Б. Репнікова, Л. Ю. Юрчук., Ю.С. Тимофєєва. – Київ : КПІ ім. Ігоря Сікорського, 2020. –78 с.

ДОДАТОК А

Блок-схема алгоритму програми мікрокомп'ютера



ДОДАТОК Б

Код програми мікрокомп'ютера

```
import RPi.GPIO as GPIO

import pyrebase

import datetime

from time import sleep

from picamera import PiCamera


GPIO.setmode(GPIO.BCM)

GPIO.setup(23, GPIO.IN)


config = {

  "apiKey": "RWkpb8AIzAxHAlB6- oIuwJJ_Gna mJvaHSy-YPqU",

  "authDomain": "home-security-2gxd5.firebaseapp.com",

  "databaseURL": "https://home-security-2gxd5.firebaseio.com",

  "storageBucket": "home-security-2gxd5.appspot.com"

}


firebase = pyrebase.initialize_app(config)

db = firebase.database()

storage = firebase.storage()
```

```
camera = PiCamera()
camera.resolution = (1280, 720)
camera.rotation = 180


# Generate file name based on current time and needed media type
def create_file_name(media_type):
    filename_suffix = datetime.datetime.now().strftime("%d%m%y_%H%M")
    file_type = ".jpg" if (media_type == "IMAGE") else ".h264"
    file_name = "_".join([media_type, filename_suffix]) + file_type
    return file_name


# Generate cloud path based on file name and needed media type
def create_cloud_path(media_type, file_name):
    storage_path = "/image/" if (media_type == "IMAGE") else "/video/"
    return storage_path + file_name


# Create media file and upload it on cloude storage
def create_and_uload_media(media_type):
    # New file name
    file_name = create_file_name(media_type)
    # Local and cloud file paths
```

```
local_path = "/home/pi/Desktop/Storage/" + file_name

cloud_path = create_cloud_path(media_type, file_name)


# Start camera preview

camera.start_preview(alpha=100)


# Get image or video from camera

if (media_type == "IMAGE"):

    sleep(3)

    camera.capture(local_path)

else:

    camera.start_recording(local_path)

    sleep(7)

    camera.stop_recording()


# Stop camera preview

camera.stop_preview()


# Upload file to Firebase storage

storage.child(cloud_path).put(local_path)
```

```
    # Return file cloud path
    return cloud_path


while True:
    # Force camera image request flag
    force_request_flag = db.child("force/flag").get().val()


    # Actual system state
    system_state = db.child("settings/state").get().val()


    if (force_request_flag):
        print("Force camera image request!")


        media_type = db.child("force/media").get().val()
        file_cloud_path = create_and_uload_media(media_type)
        msg = { "type": "force", "message": "Force image from camera", "file_path": file_cloud_path }
        db.child("events").push(msg)
        db.child("force").update({"flag": False})
        sleep(2)


    if (system_state == "ON"):
```

```
        print("System state: ON")


        sensor = GPIO.input(23)
        if (sensor == 1):
            print("Motion detected!")


            media_type = db.child("settings/media").get().val()
            file_cloud_path = create_and_uload_media(media_type)
            msg = { "type": "default", "message": "Warning! Motion detected!",
"file_path": file_cloud_path }
            db.child("events").push(msg)
            sleep(2)
        elif (sensor == 0):
            print("Keep calm...")
            sleep(2)


    else:
        print("System state: OFF")
        sleep(2)
```

ДОДАТОК В

Код функції відправки повідомлень

```
// The Cloud Functions for Firebase SDK to create Cloud Functions and setup triggers.

const functions = require('firebase-functions');

// The Firebase Admin SDK to access the Firebase Realtime Database.

const admin = require('firebase-admin');

admin.initializeApp();

exports.sendmessage = functions.database.ref('/events/{eventId}')

.onCreate((snapshot, context) => {


const bodyPath = snapshot.val().file_path;

const bodyText = snapshot.val().message;

const bodyType = snapshot.val().type


const payload = {

    notification: {

        title: 'Home security',

        body: JSON.stringify({

                path: bodyPath,

                text: bodyText,
```

```
                type: bodyType
        })
    }
};

const options = { priority: "high" };

return admin
    .database()
    .ref('auth/token')
    .once('value')
    .then(dataSnapshot => {
      const token = dataSnapshot.val();
      return admin.messaging().sendToDevice(token, payload, options);
    })
    .catch(error => {
      console.log('Error sending message:', error);
      return false;
    });
});
```

ДОДАТОК Г

Код служби інформування в мобільному застосунку

```
class CloudMessagingService : FirebaseMessagingService() {

    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        showNotification(remoteMessage.notification)
    }

    override fun onNewToken(token: String) {
        super.onNewToken(token)
    FirebaseDatabase.getInstance().getReference("auth/token").setValue(token)
    }

    private fun showNotification(message: RemoteMessage.Notification?) {
        val notificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
        val NOTIFICATION_CHANNEL_ID = "appstudio.maxdev.homesecurity"
//your app package name
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val notificationChannel = NotificationChannel(
                NOTIFICATION_CHANNEL_ID, "Notification",
                NotificationManager.IMPORTANCE_DEFAULT
            )
            notificationChannel.enableLights(true)
            notificationChannel.lightColor = Color.BLUE
            notificationChannel.vibrationPattern = longArrayOf(0, 1000, 500, 1000)
            notificationManager.createNotificationChannel(notificationChannel)
        }
```

```
        val notificationBuilder = NotificationCompat.Builder(this,
NOTIFICATION_CHANNEL_ID)
        val bodyModel = Gson().fromJson(message?.body,
NotificationBodyModel::class.java)

        val notifyIntent = Intent(this, MainActivity::class.java).apply {
            flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
            putExtra(PATH_EXTRA, bodyModel.path)
        }
        val notifyPendingIntent = PendingIntent.getActivity(
            this, 0, notifyIntent, PendingIntent.FLAG_UPDATE_CURRENT
        )
        notificationBuilder.setAutoCancel(true)
            .setDefaults(Notification.DEFAULT_ALL)
            .setWhen(System.currentTimeMillis())
            .setSmallIcon(R.drawable.ic_home_security_logo)
            .setContentTitle(message?.title)
            .setContentText(bodyModel.text)
            .setContentIntent(notifyPendingIntent)
        notificationManager.notify(Random().nextInt(), notificationBuilder.build())
    }
}
```